**Full Abstraction for
Nondeterministic Dataflow Networks**

James R. Russell*

TR 89-1022
June 1989

Department of Computer Science
Cornell University
Ithaca, NY 14853-7501

*This research was supported in part by an NSF graduate fellowship.

Full Abstraction for Nondeterministic Dataflow Networks

James R. Russell*

Computer Science Department, Cornell University
Ithaca, NY 14853

June 30, 1989

Abstract

We discuss the problem of finding fully abstract semantic models for nondeterministic dataflow networks. We present the result that there exist nondeterministic networks using only bounded choice for which the input-output relation is not compositional. We go on to show that the trace semantics is fully abstract for *all* nondeterministic as well as deterministic networks.

1 Introduction

The well-known Kahn's principle gives a fully abstract semantic model for determinate dataflow networks. In this paper we consider the problem of extending Kahn's principle by developing fully abstract semantic models for nondeterministic dataflow networks. We present the result that the straightforward generalization of Kahn's principle fails for even the weakest type of nondeterministic networks. We go on to show that a trace semantics is fully abstract for the full range of nondeterministic and deterministic networks, providing a generalized framework for describing the semantics of dataflow networks.

Dataflow networks are an important model for asynchronous parallel computation, in which concurrently executing processes communicate by sending streams of tokens along FIFO channels. Recently, there has been much work on the problem of finding semantic models for various kinds of indeterminate dataflow networks. Semantic models are mathematical descriptions of operational behavior, and are important to simplify

*Supported in part by an NSF graduate fellowship

reasoning about computations of networks of processes. A key feature of dataflow networks is that they can be composed, and one network used as a component in constructing another. We therefore require of any semantics that it be *compositional*, i.e. that the denotation of a network can be constructed from the denotations of its components. This property of a semantic model is also called computational adequacy, since it means that processes are distinguished denotationally if they are observably different in some network context. A further desirable property of a semantics is *full abstraction*, in which the converse of adequacy holds as well; i.e. two processes are distinguished denotationally if and only if they are observably different in some network context. A fully abstract semantics can intuitively be thought of as capturing the minimal amount of information necessary to obtain compositionality.

Dataflow networks with only determinate processes were first studied by Kahn [Kah77], who gave an elegant semantic model of this case. In his semantics, the denotations of processes are by stream valued functions corresponding to the input-output behavior of the process, and composition of network elements is simply function composition. This semantics is fully abstract, largely because the input-output relations (which is the observable behavior) of determinate networks are compositional.

In the case of indeterminate dataflow networks, the situation is not so simple. Brock and Ackerman [BA81] showed that for networks containing the nondeterministic primitive *fair merge* the input-output relations are not compositional, hence a straightforward generalization of Kahn's principle to these indeterminate networks will fail. Therefore any compositional model for these networks must include more information than just the observable input-output relation. The indeterminate situation is further complicated by the recent discoveries of Panangaden, Stark, and others [MPS88,Sta88,PS88,PS87] that there is in fact a hierarchy of several levels of provably inequivalent nondeterministic primitives, ranging over both bounded and unbounded indeterminacy. The strongest of the nondeterministic primitives is fair merge, and the weakest is ordinary nondeterministic 'or' (or equivalently, *unfair merge*). Such primitives as *angelic merge* and *infinity fair merge* fall into distinct intermediate levels.

Recent work by Kok, Jonsson, Panangaden and Shanbhogue [Kok88,Jon89,PS89] has succeeded in solving the problem uncovered by Brock and Ackerman, and demonstrated that a semantic model based on traces is fully abstract for networks containing fair merge. However, the proofs of full abstraction make explicit use of a fair merge process, and hence are not valid for lower levels of the nondeterministic hierarchy.

Several questions remain: Do anomalies like that shown by Brock and Ackerman exist

at lower levels of the nondeterministic hierarchy, or will a semantics like Kahn’s based on the input-output relations be fully abstract at some level below fair merge? What kind of semantic models are fully abstract for weaker forms of nondeterminism? Is there a semantic model that is fully abstract for the whole hierarchy?

The answers to these questions are the main results of this paper.

2 Dataflow networks

2.1 Background and Definitions

In this section we briefly review the definitions and terminology of dataflow networks. Since the main point of this paper is to investigate relations between semantic models and observable properties of networks, we only give an informal presentation of the main concepts. See, for example, [Sta89,PS88,Sta87,JK88,Jon89,PS89] for the formal development on which this is based.

The fundamental unit of a network is an *input-output port automaton*. These automata communicate with each other and the outside world by sending and receiving data values on “ports”. Each port is either an input or an output port for the automaton, and in each step of its execution an automata may poll or read an input port, write to an output port, or change its internal state (do internal computations).

A *dataflow network* consists of a set of concurrently executing port automata connected together by directed channels. The channels act as perfect, unbounded FIFO queues. Each channel may be connected to at most one input port, and at most one output port. There are three types of channels: *input channels*, which are not connected to an output port of any automaton, and transmit data into the network from outside; *output channels*, which are not connected to an input port of any automaton, and transmit data from the network outside; and *internal channels*, which transmit data between network nodes.

An important feature of dataflow networks is that they can be composed, and larger networks built using smaller networks in place of individual automata. As with the port automata, networks to be composed must satisfy the compatibility condition that each channel may occur at most once as an input channel and at most once as an output channel. The definitions of input and output channels remain the same (though in the process of composition an external channel of a component network may become an internal channel of the larger network). For a network N , we denote its set of input channels by I_N and its set of output channels by O_N .

A *computation* of a network is a sequence of state transitions of the component automata. We define *communication events* as transitions of a computation in which data either arrives on an input channel or is sent along an output channel (*input events* and *output events*, respectively). A *trace* of a network is a sequence of communications events on the external channels of a network. Traces are commonly written as sequences of pairs $\langle \text{channel_name}, \text{data_value} \rangle$. For a network N , we define $\Gamma[N]$ as the set of computations of N , and $T[N]$ as the set of traces of N . If Γ is a computation of N , and C is a set of channels, we define $\Gamma|_C$ as the subsequence of Γ consisting only of events on channels in C . Similarly, for a trace T of N , we define $T|_C$ as the subsequence of T consisting only of events on channels in C .

We call the sequence of values of the input or output events on a channel the *history* of the channel. We write $H_c(\Gamma)$ for the sequence of values passed over the channel c in the computation Γ – the history of channel c in the computation. Similarly, for a trace T we define $H_c(T)$ as the sequence of values from all the events on channel c in T . For a set of channels C , we define $H_C(\Gamma)$ (similarly $H_C(T)$) as the C -indexed tuple of histories of the channels in C .

2.2 Operational Semantics

The *input-output relation* of a network N is the set of all pairs of input and output channel histories, $(H_{I_N}(\Gamma), H_{O_N}(\Gamma))$, with $\Gamma \in \Gamma[N]$. The input-output relation of a network is what we consider to be the “observable” behavior. From the outside, an observer can see the values of the input and output channels, but cannot distinguish the relative order of the input events, output events, and internal events. Note that we consider the full, possibly infinite, streams of values to be observable. Other, more restrictive notions are possible, as in the work of Rabinovich and Trakhtenbrot [RT88], who only consider finite observations. We write $\mathcal{IO}[N]$ for the input-output relation of the network N , and take it as our *operational semantics*.

We now define properties useful in relating abstract semantics to observable operational behavior. We say two networks N_1 and N_2 are *operationally equal* if, for every network context $\mathcal{C}[\]$, the composite networks $\mathcal{C}[N_1]$ and $\mathcal{C}[N_2]$ have the same input-output relation, i.e. that $\mathcal{IO}[\mathcal{C}[N_1]] = \mathcal{IO}[\mathcal{C}[N_2]]$.

A semantic model $\mathcal{D}[\]$ is *adequate* if whenever $\mathcal{D}[N_1] = \mathcal{D}[N_2]$, N_1 and N_2 are operationally equal. A semantic model $\mathcal{D}[\]$ is *fully abstract* if the converse of adequacy holds as well, i.e. that $\mathcal{D}[N_1] = \mathcal{D}[N_2]$ if and only if N_1 and N_2 are operationally equal.

3 Indeterminacy and Trace Semantics

The first major work in the area of dataflow semantics was by Kahn [Kah77], who gave a simple and elegant semantics for dataflow networks in which all the processes are determinate. His semantics describes networks as continuous stream valued functions corresponding to the (functional) input-output relation of the network. This semantics is fully abstract, since it coincides with the operational semantics (the input-output relation), and the input-output relation is compositional for determinate networks.

Indeterminate dataflow networks are those for which the input-output relation is not functional. The most common examples of indeterminate networks are those containing the various *merge* primitives, e.g. fair merge, angelic merge, infinity-fair merge, and unfair merge. While for some time identifying such networks simply as indeterminate was assumed to have an unambiguous meaning, the work of Panangaden, Stark, and others [MPS88,Sta88,PS88,PS87] has shown that in fact there are many provably inequivalent indeterminate primitives. We can relate indeterminate primitives by the class of operationally distinct networks constructible using them, and they have shown that when related in this way they form a hierarchy of several different levels. In particular, each of the merge primitives mentioned above is in a distinct level of this nondeterministic hierarchy. For example, for any network constructed using the angelic merge primitive, there is an operationally equivalent network constructed using fair merge, but the converse is not true. Fair merge represents the highest known level of the nondeterministic hierarchy, bounded choice (which is equivalent to unfair merge) represents the lowest level, and there are more intermediate levels than those for the other merge primitives.

The paper by Brock and Ackerman [BA81] showed that the problem of distinguishing the behavior of indeterminate networks is more complicated than in the determinate case. They demonstrated that for networks with fair merge the input-output relation is not compositional, by exhibiting two processes with the same input-output relation that were not operationally equal (i.e. they exhibited a network context such that when composed with each of the processes, had different input-output relations). Thus, any compositional semantic model for such networks must embody more information than just the input-output relation. Since this discovery, many compositional models of indeterminacy have been developed [Pan85,BA81,KP85,Bro83,Pra86,Par82], but only recently have fully abstract models emerged [PS89,Jon89,JK88,Kok88].

Two recent papers [PS89,Jon89] have shown compositionality and full abstraction for a semantics based on trace sets $\mathcal{T}[\![\cdot]\!]$. However, in the proofs of full abstraction, they rely

on the use of a fair merge node to operationally distinguish networks with different trace sets. Thus, the full abstraction is only guaranteed for networks at the highest level of the hierarchy, those containing fair merge. The adequacy of the trace semantics holds for all networks, since from the definition it is clear that adequacy at the top level implies adequacy at all lower levels. In another paper on full abstraction, Kok [Kok88] uses a different formalism and shows full abstraction for networks with angelic merge (the next-to-highest level of the hierarchy).

While this work solves the problem exposed by Brock and Ackerman, it leaves larger unresolved questions: Is the Brock-Ackerman anomaly a phenomenon of only the highest levels of the nondeterministic hierarchy or do similar anomalies arise at lower levels? What kinds of semantic models are fully abstract for networks using primitives from lower levels of the hierarchy? Is there a model that is fully abstract for all indeterminate and determinate dataflow networks?

We supply the answers to these questions in the following sections.

4 An Example for Bounded Nondeterminism

In this section we give an example which shows that the input-output relation of networks is not compositional in the presence of even the weakest form of nondeterminism – bounded choice. This extends the scope of the Brock-Ackerman anomaly to the entire nondeterministic hierarchy.

Theorem 1. The input-output relation is not compositional for networks containing bounded choice.

Proof: Consider the two network processes each with one input channel and one output channel shown in figure 1. The operation of these processes is as follows: P_1 nondeterministically chooses between the two following determinate behaviors:

- A) First read a token from channel a , then output a 0 followed by a 1 on channel b , and stop.
- B) First output a 0 on b , then read a token from a , then output another 0 on b , and stop.

P_2 chooses among A) and B) above as well as:

- C) First output a 0 on b , then read a token from a , then output a 1 on b , and stop.

Figure 1: The example processes

The above reads are determinate, meaning that they do not time out, and that if channel a is always empty, the process waits forever.

Note that P_1 and P_2 have the same input-output relations, given by $\mathcal{IO}\llbracket P_1 \rrbracket = \mathcal{IO}\llbracket P_2 \rrbracket = \{(\epsilon, \epsilon), (\epsilon, 0)\} \cup \{(t, 01), (t, 00) \mid t \text{ any nonempty stream}\}$ (we use ϵ to denote the empty stream). However, if we look at the networks obtained by composing with the process $Copy$, which copies all tokens from channel b to channels a and c , we see that they are observably different. Specifically, their input-output relations are given by $\mathcal{IO}\llbracket Copy[P_1] \rrbracket = \{(\epsilon, \epsilon), (\epsilon, 00)\}$ and $\mathcal{IO}\llbracket Copy[P_2] \rrbracket = \{(\epsilon, \epsilon), (\epsilon, 00), (\epsilon, 01)\}$ (since the composite network has no input channels, the input part of the input-output pairs is nonexistent). ■

Though simple to describe, this is an important example – it shows that we must look for a semantic model more detailed than Kahn’s in order to describe networks with any kind of nondeterminism.

While bounded choice may be viewed as the weakest form of purely “internal” nondeterminism, recent work of Rabinovich and Trakhtenbrot [RT89] has defined a weaker type in the form of “unambiguous nonfunctional processes”. It is worth noting here that even for networks containing this more restrictive nondeterminism the input-output relation is not compositional, though the example for this case is more complex.

5 Full Abstraction

In this section we extend the work in [PS89, Jon89, JK88] and show that the semantics in which a network N is represented by its set of traces $\mathcal{T}\llbracket N \rrbracket$ is fully abstract for all

networks, with no restriction on the class of primitives used. In these papers the authors have shown $\mathcal{T}[\![\]\!]$ to be adequate, and described the closure properties and composition rules for $\mathcal{T}[\![N]\!]$. We do not repeat these descriptions here.

We will show $\mathcal{T}[\![\]\!]$ to be fully abstract by demonstrating that if $\mathcal{T}[\![N]\!] \neq \mathcal{T}[\![M]\!]$ then there is a network context $\mathcal{C}[\![\]\!]$ such that $\mathcal{IO}[\![\mathcal{C}[N]]\!] \neq \mathcal{IO}[\![\mathcal{C}[M]]\!]$. In fact, we construct such a context that is independent of N and M (except for the number of input and output channels). Furthermore, the context we construct is composed only of determinate (and sequential) processes, and therefore puts no restriction on the class of networks for which $\mathcal{T}[\![\]\!]$ is fully abstract. We construct this context by borrowing from ideas common in complexity theory and verifying (rather than generating) traces.

Consider the network context $\mathcal{V}[\![\]\!]$ shown in figure 2. $\mathcal{V}[\![\]\!]$ has $I_{\mathcal{V}[N]} = \{Fin\}$ and

Figure 2: The context $\mathcal{V}[\![\]\!]$

$O_{\mathcal{V}[N]} = \{s, out''_1, \dots, out''_l\}$, and is composed of the following determinate processes:

Feeder: This expects a trace of N as input on the channel Fin , which it reads one event at a time. For each input event $\langle in_j, d \rangle$ it gets on Fin , it sends the token d out on channel in_j . For each output event $\langle out_j, d \rangle$ it gets on Fin , it sends this pair out on channel $OVin$, and then waits until it receives the distinguished signal token $\#$ on channel s' before continuing. This is a determinate process, so if any input it expects is malformed or fails to arrive, it halts. The important aspects of *Feeder* are that it supplies tokens to the channels in_j in the same order as they appear in its input, and its output to the channel $OVin$ is exactly the subsequence of its input consisting of output event pairs.

Output Verifier: This reads a sequence of output events off the channel $OVin$. For

each event $\langle out_j, d \rangle$ it receives, it waits until it can read a token d' on channel out'_j , and if $d = d'$, it sends a signal token $\#$ out on channel s , then channel s' . If $d \neq d'$ it halts.

C_j : These are copy nodes. They read from channel out_j and copy onto out'_j , then out''_j .

N : This can be any network with $I_N = \{in_1, \dots, in_k\}$ and $O_N = \{out_1, \dots, out_l\}$. N is not required to be determinate.

We now prove that the context $\mathcal{V}[\]$ actually “verifies” traces of N . Given a trace T , define the string $sig_string_T \in \{\#\}^* \cup \{\#\infty\}$ such that $|sig_string_T| = |T|_{O_N}$, i.e. sig_string_T has exactly one distinguished signal token $\#$ for each output event in T . This string serves to tell the *Feeder*, and an external observer, that the output events of T were actually observed.

Lemma 1. A trace T is a possible trace of the network N if and only if on input T to the network $\mathcal{V}[N]$, a possible output is sig_string_T on channel s and $H_{out_j}(T)$ on channel out_j for $1 \leq j \leq l$. That is, $T \in \mathcal{T}[N] \iff (T, \langle sig_string_T, H_{O_N}(T) \rangle) \in \mathcal{IO}[\mathcal{V}[N]]$.

Proof: $\implies$: We will use the notation T_n to denote the prefix of the trace T consisting of the first n events of T . First, we note the following fact: If there exists a trace $S \in \mathcal{T}[N]$ such that

- (i) $H_{I_N}(S) = H_{I_N}(T)$ & $H_{O_N}(S) = H_{O_N}(T)$, and
- (ii) $\forall n \exists m$ s.t. $H_{I_N}(S_m) = H_{I_N}(T_n)$ & $H_{O_N}(S_m) \sqsubseteq H_{O_N}(T_n)$,

then we may conclude $T \in \mathcal{T}[N]$, since S may be transformed into it by delaying output events. This relies on the fact that $\mathcal{T}[N]$ is closed under such transformations. We will show that under the hypothesis, such a trace S must exist.

Let Γ be a computation of $\mathcal{V}[N]$ such that $H_{I_{\mathcal{V}[N]}}(\Gamma) = T$, and $H_{O_{\mathcal{V}[N]}}(\Gamma) = \langle sig_string_T, H_{O_N}(T) \rangle$. Since Γ includes a computation of N , we know $\Gamma|_{I_N \cup O_N}$ is in $\mathcal{T}[N]$. Let $S = \Gamma|_{I_N \cup O_N}$. It follows from the definition of *Feeder* that $\Gamma|_{I_N} = T|_{I_N}$, and hence $H_{I_N}(S) = H_{I_N}(T)$. By assumption, we have $H_{\{out'_1, \dots, out'_l\}}(\Gamma) = H_{O_N}(T)$, so from the definition of the C_i , we know that $H_{O_N}(\Gamma) = H_{O_N}(S) = H_{O_N}(T)$.

Now, for all n we define Γ_n as the largest prefix of Γ such that $\Gamma_n|_{I_N} = T_n|_{I_N}$, and $H_{O_{\mathcal{V}[N]}}(\Gamma_n)|_{O_N} = T_n|_{O_N}$. Let $S_{m(n)} = \Gamma_n|_{I_N \cup O_N}$. Then we see that

$$\begin{aligned} \Gamma_n|_{I_N} = T_n|_{I_N} &= H_{I_N}(\Gamma_n) = H_{I_N}(T_n) \\ &= H_{I_N}(S_{m(n)}) = H_{I_N}(T_n). \end{aligned}$$

From the definition of *Output Verifier* and the maximality of Γ_n , we know that every output event on $OVin$ is matched by an event on an out' channel. Hence, for $1 \leq j \leq l$,

$$\begin{aligned} H_{OVin}(\Gamma_n) \upharpoonright_{out_j} \sqsubseteq \Gamma_n \upharpoonright_{out'_j} &= H_{O_N}(H_{OVin}(\Gamma_n)) \sqsubseteq H_{\{out'_1, \dots, out'_l\}}(\Gamma_n) \\ &= H_{O_N}(T_n) \sqsubseteq H_{\{out'_1, \dots, out'_l\}}(\Gamma_n) \text{ (by def'n of Feeder)} \\ &= H_{O_N}(T_n) \sqsubseteq H_{O_N}(\Gamma_n) = H_{O_N}(S_{m(n)}) \text{ (by def'n of } C_i) \end{aligned}$$

Now, by applying the fact stated above, we conclude that $T \in \mathcal{T}\llbracket N \rrbracket$.

$\implies$: Assume $T \in \mathcal{T}\llbracket N \rrbracket$. We show how to construct compatible traces of the components of $\mathcal{V}\llbracket N \rrbracket$, such that their composition, T^* (which is a trace of $\mathcal{V}\llbracket N \rrbracket$), generates the input-output pair

$$(H_{I_{\mathcal{V}\llbracket N \rrbracket}}(T^*), H_{O_{\mathcal{V}\llbracket N \rrbracket}}(T^*)) = (T, \langle sig_string_T, H_{O_N}(T) \rangle).$$

We construct the traces for each component based on T as follows:

Feeder: For each event $\langle in_j, d \rangle$ in T , replace it with the events $\langle Fin, \langle in_j, d \rangle \rangle \langle in_j, d \rangle$. Replace each event $\langle out_j, d \rangle$ with the events $\langle Fin, \langle out_j, d \rangle \rangle \langle OVin, \langle out_j, d \rangle \rangle \langle s', \# \rangle$.

Output Verifier: Leave out all events in T of the form $\langle in_j, d \rangle$, and replace each event $\langle out_j, d \rangle$ by the events $\langle OVin, \langle out_j, d \rangle \rangle \langle out'_j, d \rangle \langle s', \# \rangle \langle s, \# \rangle$.

C_i : For each event of the form $\langle out_i, d \rangle$ in T , replace it by the events $\langle out_i, d \rangle \langle out'_i, d \rangle \langle out''_i, d \rangle$. Leave out all other events in T .

It is straightforward to show that the traces constructed above and T satisfy the technical definition of compatibility (this definition is not given here, but the network behavior is intuitive from the definitions). Their composition yields the trace T^* defined as follows: For each event $\langle in_j, d \rangle$ in T , replace it with $\langle Fin, \langle in_j, d \rangle \rangle$, and replace each event $\langle out_j, d \rangle$ with the events $\langle out''_j, d \rangle \langle s, \# \rangle$. Clearly, T^* is in $\mathcal{T}\llbracket \mathcal{V}\llbracket N \rrbracket \rrbracket$, $H_{I_{\mathcal{V}\llbracket N \rrbracket}}(T^*) = T$, and $H_{O_{\mathcal{V}\llbracket N \rrbracket}}(T^*) = \langle sig_string_T, H_{O_N}(T) \rangle$. Hence, $(T, \langle sig_string_T, H_{O_N}(T) \rangle) \in \mathcal{IO}\llbracket \mathcal{V}\llbracket N \rrbracket \rrbracket$. ■

From the above lemma, the full abstraction of $\mathcal{T}\llbracket \cdot \rrbracket$ follows directly.

Theorem 2 (Full Abstraction) Given dataflow networks N and M , $\mathcal{T}\llbracket N \rrbracket = \mathcal{T}\llbracket M \rrbracket$ if and only if $\mathcal{IO}\llbracket \mathcal{C}\llbracket N \rrbracket \rrbracket = \mathcal{IO}\llbracket \mathcal{C}\llbracket M \rrbracket \rrbracket$ for all network contexts $\mathcal{C}[\cdot]$.

Proof: We already know that $\mathcal{T}\llbracket \cdot \rrbracket$ is adequate – this is the 'only if' direction. For the other direction, assume there exists $T \in \mathcal{T}\llbracket N \rrbracket, T \notin \mathcal{T}\llbracket M \rrbracket$. Then by the above lemma, the input-output pair $(T, \langle sig_string_T, H_{O_N}(T) \rangle)$ is in $\mathcal{IO}\llbracket \mathcal{V}\llbracket N \rrbracket \rrbracket$ but not $\mathcal{IO}\llbracket \mathcal{V}\llbracket M \rrbracket \rrbracket$, so $\mathcal{IO}\llbracket \mathcal{V}\llbracket N \rrbracket \rrbracket \neq \mathcal{IO}\llbracket \mathcal{V}\llbracket M \rrbracket \rrbracket$. ■

6 Conclusions and Future Work

In the above we have shown that the straightforward generalization of Kahn's semantics will fail for any nondeterministic dataflow network. We have also proved fully abstract a semantics for all nondeterministic and deterministic dataflow networks, thereby providing a model considerably more general than Kahn's. In a future paper we will present a generalization of the other important aspect of Kahn's principle, the iterative process by which the meaning of a network can be determined from the meanings of its components.

Rabinovich and Trakhtenbrot [RT88] have studied variations of Kahn's principle using a different notion of observation than ours, in which only finite streams are observable. Another interesting direction for future work is to extend the results in this paper to this somewhat different setting.

References

- [BA81] J. D. Brock and W. B. Ackerman. Scenarios: A model of non-determinate computation. In *Formalization of Programming Concepts*, pages 252–259, 1981. LNCS 107.
- [Bro83] M. Broy. Fixed point theory for communication and concurrency. In *Formal Description of Programming Concepts II*, pages 125–148. North-Holland, 1983.
- [JK88] B. Jonsson and J. Kok. Comparing dataflow models. Manuscript, 1988.
- [Jon89] B. Jonsson. A fully abstract trace model for dataflow networks. In *Proceedings of the Sixteenth Annual ACM Symposium On Principles Of Programming Languages*, 1989.
- [Kah77] G. Kahn. The semantics of a simple language for parallel programming. In *Information Processing 74*, pages 993–998. North-Holland, 1977.
- [Kok88] J. Kok. Dataflow semantics. Technical Report CS-R8835, Centre for Mathematics and Computer Science, August 1988.
- [KP85] R. M. Keller and P. Panangaden. Semantics of networks containing indeterminate operators. In *Proceedings of the 1984 CMU Seminar on Concurrency*, pages 479–496, 1985. LNCS 197.
- [MPS88] D. McAllester, P. Panangaden, and V. Shanbhogue. Nonexpressibility of fairness and signaling. In *Proceedings of the 29th Annual Symposium of Foundations of Computer Science*, 1988.

- [Pan85] P. Panangaden. Abstract interpretation and indeterminacy. In *Proceedings of the 1984 CMU Seminar on Concurrency*, pages 497–511, 1985. LNCS 197.
- [Par82] D. Park. The “fairness problem” and non-deterministic computing networks. In *Proceedings of the Fourth Advanced Course on Theoretical Computer Science, Mathematisch Centrum*, pages 133–161, 1982.
- [Pra86] V. Pratt. Modeling concurrency with partial orders. *International Journal Of Parallel Programming*, 15(1):33–71, 1986.
- [PS87] P. Panangaden and V. Shanbhogue. On the expressive power of indeterminate primitives. Technical Report 87-891, Cornell University, Computer Science Department, November 1987.
- [PS88] P. Panangaden and E. W. Stark. Computations, residuals and the power of indeterminacy. In Timo Lepisto and Arto Salomaa, editors, *Proceedings of the Fifteenth ICALP*, pages 439–454. Springer-Verlag, 1988. Lecture Notes in Computer Science 317.
- [PS89] P. Panangaden and V. Shanbhogue. Traces are fully abstract for networks with fair merge. unpublished manuscript, 1989.
- [RT88] A. Rabinovich and B. A. Trakhtenbrot. Nets of processes and dataflow. To appear in Proceedings of ReX School on Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency, LNCS, 1988.
- [RT89] A. Rabinovich and B. A. Trakhtenbrot. Nets and data flow interpreters. In *Proceedings of the Fourth Annual IEEE Symposium on Logic in Computer Science*, pages 164–174, 1989.
- [Sta87] E. W. Stark. Concurrent transition system semantics of process networks. In *Proceedings Of The Fourteenth Annual ACM Symposium On Principles Of Programming Languages*, pages 199–210, 1987.
- [Sta88] E. W. Stark. On the relations computable by a class of concurrent automata. Manuscript in preparation, 1988.
- [Sta89] E. W. Stark. Concurrent transition systems. *Theoretical Computer Science*, 1989.