On Oraclizable Networks and Kahn's Principle*

James R. Russell**

TR 89-1046
October 1989

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

*This paper will appear in the Proceedings of the 17th Annual ACM Symposium on Principles of Programming Languages, January 1990.
**Supported in part by an NSF graduate fellowship, NSF grant CCR-8818979, and an IBM graduate fellowship.

On Oraclizable Networks and Kahn's Principle*

James R. Russell†
Computer Science Department, Cornell University
Ithaca, NY 14853

October 17, 1989

Abstract

In this paper we investigate generalizations of Kahn's principle to nondeterministic dataflow networks. Specifically, we show that for the class of "oraclizable" networks a semantic model in which networks are represented by certain sets of continuous functions is fully abstract and has the fixed-point property. We go on to show that the oraclizable networks are the largest class representable by this model, and are a proper superclass of the networks implementable with the *infinity fair merge* primitive. Finally, we use this characterization to show that infinity fair merge networks and oraclizable networks are proper subclasses of the networks with Egli-Milner monotone input-output relations.

1 Introduction

Dataflow networks are an important model for asynchronous parallel computation, in which concurrently executing processes communicate by sending streams of tokens along FIFO channels. The well known Kahn's principle [Kah77] gives a semantic model for determinate dataflow networks in which the networks are represented by continuous stream valued functions. This model has the two desirable properties that it is fully abstract and that the denotation of a composite network can be computed from the denotations of its components via a fixed-point construction. In this paper we investigate generalizations of Kahn's principle to nondeterministic dataflow networks.

Recently, there has been much work on the problem of finding semantic models generalizing Kahn's to various kinds of indeterminate dataflow networks. The most common examples of indeterminate networks are those containing the various *merge* primitives,

*This paper will appear in the Proceedings of the 17^{th} Annual ACM Symposium on Principles of Programming Languages, January 1990.

†Supported in part by an NSF graduate fellowship, NSF grant CCR-8818979, and an IBM graduate fellowship.

e.g. fair merge, angelic merge, infinity-fair merge, and unfair merge. Recent work of Panangaden, Stark, and others [MPS88,Sta88,PS88,PS87] has shown that these primitives have provably inequivalent expressive power. In particular, they have shown that these primitives form a hierarchy of expressibility, with fair merge at the highest level.

Many semantic models for indeterminate networks have been developed [Pan85,BA81, KP85,Bro83,Pra86,Par82], and only recently have fully abstract models emerged [PS89, Jon89,JK88,Kok88,Rus89]. However, these fully abstract models are based on traces or similar formalisms, and do not have a fixed-point theory. Keller and Panangaden [KP86, Pan85] and Broy [Bro83] have both developed models for the full range of nondeterminism that employ fixed-point constructions, but they are cumbersome and not fully abstract. Misra [Mis89] has described a nice equational system for reasoning about nondeterministic networks in which network meanings are 'smooth' solutions to recursive equations, but he does not consider full abstraction, nor provide fixed-point techniques for computing the smooth solutions. Abramsky [Abr89] has developed a general categorical theory for Kahn-type models for indeterminate dataflow networks.

The approach we take in this paper is to look at a restricted class of nondeterministic dataflow networks, develop a model for this class that is both fully abstract and has a fixed-point theory, and to characterize the representation and expressiveness of this class. The class we consider is that of oraclizable networks. We show that for this class a semantics based on sets of continuous functions is fully abstract and has a simple and general fixed-point principle. We also show that this class is universal for the sets of functions representation, i.e. that it is the largest class describable by sets of functions.

We relate the oraclizable networks to the hierarchy of nondeterministic primitives by noting that they properly contain all networks implementable with infinity-fair merge. One characteristic that separates infinity-fair merge networks from networks at higher levels of the hierarchy is that the input-output relations of the infinity-fair merge networks are monotone in the Egli-Milner ordering, while this is not necessarily the case for networks at higher levels. It has been conjectured that infinity-fair merge is universal for Egli-Milner monotone relations, i.e. that any Egli-Milner monotone relation is the input-output relation of some network implementable with infinity-fair merge. Using the above characterization and semantics we are able to show that this conjecture is false.

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. There are two atomic operation of network composition: aggregation and looping. The aggregate of networks M and N, written $M||N$, is the network formed by combining them 'side-by-side' with no identification of channels. Given a network M, $\mathsf{loop}(a, b, M)$ is the network formed from M by identifying input channel a with output channel b. It is clear that any network can be constructed from these operations.

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 $\langle$*channel_name*, *data_value*$\rangle$, and we write $\mathcal{T}[\![N]\!]$ for the set of traces of a network N. We call the sequence of values of the events on the input (or output) channels the *input* (or *output*) *history* of a trace.

2.2 Operational Semantics

The *input-output relation* of a network N is the set of all pairs of input and output histories for traces in $\mathcal{T}[\![N]\!]$. 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 consider a theory based on 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 Kahn's Semantics and Traces

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 has the desirable properties that it is fully abstract, and the denotation of a composite network can be obtained from the denotations of its components as the least fixed point of the equations describing the network. Specifically, a network M with m input channels and n output channels is represented as a function $f_M : \mathsf{S}^m \to \mathsf{S}^n$, where S is the domain of streams. If N is a network with m' inputs and n' outputs, then $f_{M\|N} = \langle f_M, f_N \rangle : \mathsf{S}^{m+m'} \to \mathsf{S}^{n+n'}$, where $\langle f_M, f_N \rangle$ is the function that on input $\langle i, i' \rangle \in \mathsf{S}^{m+m'}$ (with $i \in \mathsf{S}^m, i' \in \mathsf{S}^{m'}$) produces output $\langle f_M(i), f_N(i') \rangle \in \mathsf{S}^{n+n'}$. Similarly, $f_{\mathsf{loop}(a,b,M)} = \mathsf{fix}(a, b, f_M)$, where $g = \mathsf{fix}(a, b, f_M)$ is the function in $\mathsf{S}^{m-1} \to \mathsf{S}^{n-1}$ that computes fixed points of f_M; $g(i) = o$ means that o is the least output on the components other than b such that there exists a stream l with $f_M(i, l) = (o, l)$ (where l is actually the ath component of the input and the bth component of the output). The importance of the semantics having this fixed-point property is that it provides us with a simple method for computing the meaning of a looped network as a limit.

Indeterminate dataflow networks are those for which the input-output relation is not functional. Obviously such networks cannot be represented by a function, and a representation by the (nonfunctional) input-output relation fails to be fully abstract [BA81,Rus89]. Thus, the naive generalizations of Kahn's semantics to the indeterminate setting fail, and different models have to be sought out. In this paper we develop such a model for the class of oraclizable nondeterministic networks.

The work in [Jon89,PS89,Rus89] shows that the trace semantics $\mathcal{T}[\![]\!]$ is fully abstract, and in the following sections we compare our semantics to $\mathcal{T}[\![]\!]$ rather than directly to the operational semantics. We also employ an alternative shorthand notation for traces which we call *checkpoint sequences*. We define a checkpoint sequence for a network M as a sequence $(i_0, o_0), (i_1, o_1), \ldots (i, o)$ where i_n and o_n are tuples of finite input and output streams (one for each channel), $(i_n, o_n) \sqsubseteq (i_{n+1}, o_{n+1})$ for all n, and $(i, o) = \sqcup(i_n, o_n)$. We regard such a checkpoint sequence as shorthand for a trace t of M consisting of all the input events of i_0, followed by the output events of o_0, followed by the input events of $i_1 - i_0$, followed by the output events of $o_1 - o_0$, etc. Furthermore the entire input history of t must be i, and the output history of t must be o. Note that a given checkpoint sequence actually represents a family of traces related by the permutation of adjacent input (or output) events on different channels, since trace sets are closed under such permutations.

4 Semantics of Oraclizable Networks

In this section we investigate semantic models that generalize Kahn's to the class of oraclizable networks. In the first subsection we define oraclizable networks and present our first semantic model. This model is a straightforward extension of Kahn's in which oraclizable networks - which can be thought of as nondeterministic networks choosing among many different determinate behaviors - are represented by *sets* of stream valued functions, corresponding to the sets of possible determinate behaviors. With this representation the fixed-point property of Kahn's semantics extends directly to this model, by applying it to each of the possible equations.

Unfortunately, this straightforward representation fails to be fully abstract. In the second subsection we modify the model of the first and represent oraclizable networks by sets of functions that are closed in a certain sense. With this modification, we show that this second semantics becomes fully abstract, and preserves the fixed-point property of the first.

4.1 The Direct Semantics

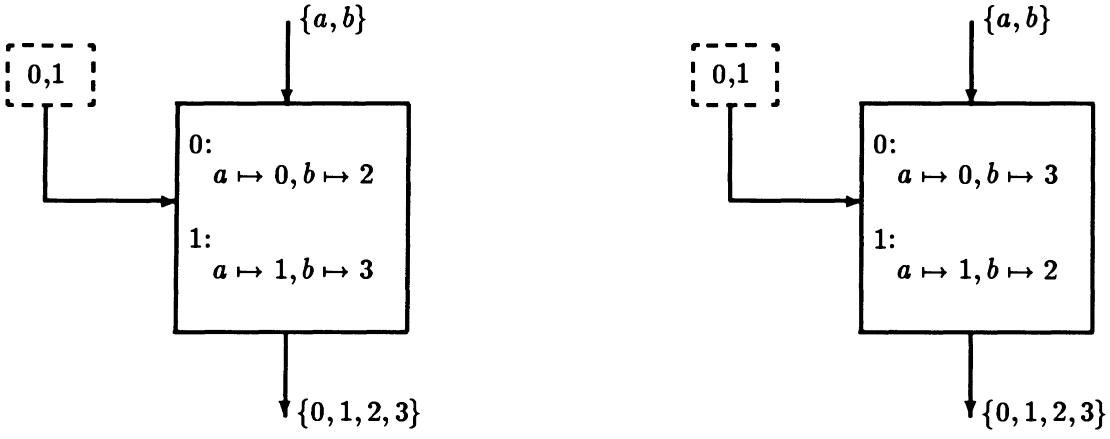
Definition 1. A nondeterministic network M is *oraclizable* if it is operationally equal to a nondeterministic network M_O without input channels (the "oracle"), connected to some inputs of a determinate network M_D.

It is easily seen that the class of oraclizable networks is closed under composition.

Given an oraclizable network M, we can identify its oracle part M_O with the set of possible outputs of M_O. Additionally, we can regard its determinate part M_D, which we know acts as a function f_{M_D} from oracle inputs and external inputs to outputs, as function from oracle inputs to functions from external inputs to outputs. This view leads naturally to our first semantic model.

Definition 2. Given an oraclizable network M, we represent it by its set of possible functional behaviors

$$\mathcal{F}_1[\![M]\!] \stackrel{\text{def}}{=} \bigcup_{o \in M_O} f_{M_D}(o).$$



Both the above processes take as input a single token, either a or b, and if it is an a produce either 0 or 1, and if it is a b produce either 2 or 3. However, the left process uses its oracle to choose between the functions $a \mapsto 0, b \mapsto 2$ and $a \mapsto 1, b \mapsto 3$, while the right process uses its oracle to choose between the two different functions $a \mapsto 0, b \mapsto 3$ and $a \mapsto 1, b \mapsto 2$.

Figure 1: Two indistinguishable processes with different sets of functions.

Lemma 1.

$$\mathcal{F}_1[\![M||N]\!] = \{g|g = \langle f, f'\rangle, f \in \mathcal{F}_1[\![M]\!], f' \in \mathcal{F}_1[\![N]\!]\}$$
$$\mathcal{F}_1[\![\mathsf{loop}(a, b, M)]\!] = \{f|f = \mathsf{fix}(a, b, f'), f' \in \mathcal{F}_1[\![M]\!]\}$$

Proof: These follow directly from the definitions.

$\mathcal{F}_1[\![]\!]$ is an attractive semantic model, since the network composition operations of aggregation and looping correspond to function aggregation and least fixed point applied to each of the functions in the representation, as we desire. Now we compare the semantics $\mathcal{F}_1[\![]\!]$ to the fully abstract trace semantics $\mathcal{T}[\![]\!]$.

Lemma 2.

$$\mathcal{T}[\![M]\!] = \{(i_0, o_0), (i_1, o_1), \ldots (i, o)|\exists f \in \mathcal{F}_1[\![M]\!], f(i) = o, f(i_n) \sqsupseteq o_n \text{ for all } n\}$$

Proof: Direct from the definition of $\mathcal{F}_1[\![M]\!]$ and the observation that for a determinate network D,

$$\mathcal{T}[\![D]\!] = \{(i_0, o_0), (i_1, o_1), \ldots (i, o)|f_D(i) = o, f_D(i_n) \sqsupseteq o_n \text{ for all } n\}$$

From lemma 2 we conclude that $\mathcal{F}_1[\![M]\!] = \mathcal{F}_1[\![N]\!]$ implies $\mathcal{T}[\![M]\!] = \mathcal{T}[\![N]\!]$, and hence $\mathcal{F}_1[\![]\!]$ is adequate. It is not, however, fully abstract, as can be seen in the example in figure 1.

4.2 A Fully Abstract Variation

Intuitively, $\mathcal{F}_1[\![M]\!]$ fails to be fully abstract because it only includes the functional behaviors explicit in M, while there may be other functions inherent in the behavior of M, though not corresponding to any single oracle value. In order to be fully abstract, the representation must identify networks differing only by such functions.

Definition 3. Given a set of functions F, $Cl(F)$ is the closure of F under the addition of functions not finitely distinguishable from F. These are the functions inherent in the behavior of F.

$$\begin{aligned} Cl(F) \stackrel{\text{def}}{=} \{f \mid\ & \forall \text{ chains of finite inputs } i_0 \sqsubseteq i_1 \sqsubseteq \cdots, i = \bigsqcup i_n, \text{ and} \\ & \forall \text{ chains of finite outputs } o_0 \sqsubseteq o_1 \sqsubseteq \cdots, f(i) = \bigsqcup o_n, o_n \sqsubseteq f(i_n) \text{ for each } n, \\ & \exists f' \in F \text{ with } f'(i) = f(i), f'(i_n) \sqsupseteq o_n \text{ for each } n\} \end{aligned}$$

Definition 4. Given an oraclizable network M, we represent it by the closure of its set of functional behaviors.

$$\mathcal{F}_2[\![M]\!] \stackrel{\text{def}}{=} Cl(\mathcal{F}_1[\![M]\!])$$

Clearly $\mathcal{F}_2[\![M]\!] \supseteq \mathcal{F}_1[\![M]\!]$ for any M.

As justification for the claim that the additional functions were inherent in M we have the following:

Lemma 3. Given an oraclizable network M, suppose M' is an oraclizable network whose functional behaviors are all those in $\mathcal{F}_2[\![M]\!]$ (i.e. $\mathcal{F}_1[\![M']\!] = \mathcal{F}_2[\![M]\!]$). Then M and M' are operationally indistinguishable.

Proof: We will show $\mathcal{T}[\![M']\!] = \mathcal{T}[\![M]\!]$. Using lemma 2 and the definition of $\mathcal{F}_2[\![]\!]$ we have

$$\begin{aligned} \mathcal{T}[\![M']\!] &= \{(i_0, o_0), (i_1, o_1), \ldots (i, o) \mid \exists f \in \mathcal{F}_1[\![M']\!], f(i) = o, f(i_n) \sqsupseteq o_n \text{ for all } n\} \\ &= \{(i_0, o_0), (i_1, o_1), \ldots (i, o) \mid \exists f \in Cl(\mathcal{F}_1[\![M]\!]), f(i) = o, f(i_n) \sqsupseteq o_n \text{ for all } n\} \\ &= \{(i_0, o_0), (i_1, o_1), \ldots (i, o) \mid \exists f' \in \mathcal{F}_1[\![M]\!], f'(i) = f(i) = o, f'(i_n) \sqsupseteq o_n \text{ for all } n\} \\ &= \mathcal{T}[\![M]\!] \end{aligned}$$

Thus, the sets $\mathcal{F}_2[\![M]\!]$ and $\mathcal{F}_1[\![M]\!]$ represent the same operational behavior. However, $\mathcal{F}_2[\![]\!]$ is general enough that it is fully abstract.

Theorem 1 (Full Abstraction for Oraclizable Networks) Given oraclizable networks M and N,

$$\mathcal{F}_2[\![M]\!] = \mathcal{F}_2[\![N]\!] \iff \mathcal{T}[\![M]\!] = \mathcal{T}[\![N]\!]$$

Proof: $\Leftarrow$: It is clear from lemma 2 and the definition of $\mathcal{F}_2[\![]\!]$ that

$$\begin{aligned}\mathcal{F}_2[\![M]\!] = \{f \mid\ & \forall \text{ chains } i_0 \sqsubseteq i_1 \sqsubseteq \cdots, i = \bigsqcup i_n, \text{ and} \\ & \forall \text{ chains } o_0 \sqsubseteq o_1 \sqsubseteq \cdots, f(i) = \bigsqcup o_n, o_n \sqsubseteq f(i_n) \text{ for each } n, \\ & \text{the checkpoint sequence } (i_0, o_0), (i_1, o_1), \ldots (i, f(i)) \text{ is in } \mathcal{T}[\![M]\!]\}\end{aligned}$$

Hence $\mathcal{T}[\![M]\!]$ determines $\mathcal{F}_2[\![M]\!]$, and the result follows.
$\Rightarrow$: Say $(i_0, o_0), (i_1, o_1), \ldots (i, o) \in \mathcal{T}[\![M]\!]$. Then by lemma 2 we know there exists $f \in \mathcal{F}_1[\![M]\!]$ with $f(i) = o, f(i_n) \sqsupseteq o_n$ for each n. But $f \in \mathcal{F}_1[\![M]\!] \subseteq \mathcal{F}_2[\![M]\!] = \mathcal{F}_2[\![N]\!]$, so from the above equality we conclude $(i_0, o_0), (i_1, o_1), \ldots (i, o) \in \mathcal{T}[\![N]\!]$.

Thus, $\mathcal{F}_2[\![]\!]$ is a modification of $\mathcal{F}_1[\![]\!]$ that is fully abstract. We now verify that $\mathcal{F}_2[\![]\!]$ preserves the desirable composition and fixed-point properties of $\mathcal{F}_1[\![]\!]$.

Theorem 2. $\mathcal{F}_2[\![]\!]$ has the same composition and fixed-point properties as $\mathcal{F}_1[\![]\!]$. Specifically,

$$\mathcal{F}_2[\![M || N]\!] = \{g | g = \langle f, f' \rangle, f \in \mathcal{F}_1[\![M]\!], f' \in \mathcal{F}_2[\![N]\!]\}$$
$$\mathcal{F}_2[\![\mathsf{loop}(a, b, M)]\!] = Cl(\{g | g = \mathsf{fix}(a, b, g'), g' \in \mathcal{F}_2[\![M]\!]\}).$$

Proof: The aggregation property follows directly from the definitions.
We see $\mathcal{F}_2[\![\mathsf{loop}(a, b, M)]\!] \subseteq Cl(\{g | g = \mathsf{fix}(a, b, g'), g' \in \mathcal{F}_2[\![M]\!]\})$ directly, since

$$\mathcal{F}_2[\![\mathsf{loop}(a, b, M)]\!] = Cl(\mathcal{F}_1[\![\mathsf{loop}(a, b, M)]\!]) = Cl(\{f | f = \mathsf{fix}(a, b, f'), f' \in \mathcal{F}_1[\![M]\!]\})$$

and $\mathcal{F}_1[\![M]\!] \subseteq \mathcal{F}_2[\![M]\!]$.
Finally, we show $Cl(\mathcal{F}_1[\![\mathsf{loop}(a, b, M)]\!]) \supseteq \{g | g = \mathsf{fix}(a, b, g'), g' \in \mathcal{F}_2[\![M]\!]\}$ via an intricate construction that we only outline here.

Given $g' \in \mathcal{F}_2[\![M]\!], g = \mathsf{fix}(a, b, g')$, choose chains $i_0 \sqsubseteq i_1 \sqsubseteq \cdots i$, and $o_0 \sqsubseteq o_1 \sqsubseteq \cdots o$, with $g(i_n) \sqsupseteq o_n, g(i) = o$. Now we know that $g(i_n) = p_n$ means that p_n is the least tuple of output streams such that there exists a stream l (the 'looped' input) with $g'(i_n, l) = (p_n, l)$ (where l is actually the ath component of the input and the bth component of the output).

Using approximations to the fixed point, we are able to construct chains of finites

$$(i_0, \epsilon) \sqsubseteq (i_0, l_1) \sqsubseteq \cdots (i_0, l_{k_0}) \sqsubseteq (i_1, l_{k_0+1}) \sqsubseteq \cdots (i, l)$$
$$(p_0, l_1) \sqsubseteq (p_1, l_2) \sqsubseteq \cdots (p_{k_0}, l_{k_0+1}) \sqsubseteq (p_{k_0+1}, l_{k_0+2}) \sqsubseteq \cdots (p, l)$$

with $p_{k_n} \sqsupseteq o_n \forall n$, $p = o$, $g'(i_n, l_k) \sqsupseteq (p_k, l_{k+1}) \forall n, k$, and $g'(i, l) = (p, l)$.

Then, since $g' \in \mathcal{F}_2[\![M]\!]$, there exists $f' \in \mathcal{F}_1[\![M]\!]$ with $f'(i_n, l_k) \sqsupseteq (p_k, l_{k+1}) \forall n, k$, and $f'(i, l) = (p, l)$. Finally, we let $f = \mathsf{fix}(a, b, f') \in \mathcal{F}_1[\![\mathsf{loop}(a, b, M)]\!]$, and we have $f(i_n) \sqsupseteq p_{k_n} \sqsupseteq o_n \forall n$ and $f(i) = p = o$. Hence, $g \in Cl(\mathcal{F}_1[\![\mathsf{loop}(a, b, M)]\!])$.

Thus, we have succeeded in finding what we want: A semantic model for the class of oraclizable networks that is fully abstract and that comes equipped with a fixed-point principle, thereby generalizing both aspects of Kahn's principle to this class. In the next section we investigate characterizations of this class in terms of this model and its expressibility.

5 The Universality of Oraclizable Networks

In the previous section we restricted ourselves to the class of oraclizable networks and discovered that the representation by certain sets of functions was fully abstract. In this section we show that the oraclizable networks are universal for this representation; i.e. that the oraclizable networks are the largest class of networks representable by sets of functions. We go on to show that all networks constructible using the ***infinity-fair merge*** primitive are oraclizable, and that the infinity-fair merge networks are a proper subclass of the oraclizable networks.

Finally, we use these characterizations to show that another tempting conjecture fails to hold. Namely, that although the input-output relations of infinity-fair merge networks (and of all oraclizable networks) are monotone in the Egli-Milner ordering, the class of oraclizable networks (which includes infinity-fair merge networks) is a proper subset of the Egli-Milner monotone networks.

Theorem 3. The oraclizable networks are exactly the whose behavior can be described by a set of functions

Proof: By lemmas 4 and 5.

Lemma 4. The behavior of any oraclizable network is representable by a set of functions.

Proof: This is the semantics of the previous section.

Lemma 5. Given any set of functions F, there is an oraclizable network M implements F (i.e. $\mathcal{F}_1[\![M]\!] = F$).

Proof: Given F, we explicitly construct the network M, with determinate part M_D, and oracle part M_O.

The idea behind M_D is that F can be organized into a countably branching tree indexed by infinite integer sequences. For a stream i we define the notation $[i]_n$ to denote the prefix of i of length at most n; similarly for $[f(i)]_n$. Given functions $f, f' \in F$, we write $f \equiv_n f'$ iff for all i,

$$[f([i]_n)]_n = [f'([i]_n)]_n.$$

Now we note that there are only countably many equivalence classes of F modulo $\equiv_1$. Hence we can index them by integers and denote them C_{k_1}. Similarly, for every integer

k_1, we index the equivalence classes of C_{k_1} modulo $\equiv_2$ by integers and denote them $C_{k_1 k_2}$. Proceeding in this way, we can define C_s for any finite sequence s of integers. For s infinite, C_s is the intersection of all $C_{s'}$ with s' a prefix of s, and hence is either empty or contains a single function from F. We will let S be the set of infinite sequences s for which C_s is not empty.

Now we define a function P such that given input i and a sequence of integers s, we have

$$P(s,i) \stackrel{\text{def}}{=} \begin{cases} [f([i]_n)]_n, f \in C_s & \text{if } s \text{ finite of length } n \\ f(i), f \in C_s & \text{if } s \text{ infinite and } C_s \text{ not empty} \\ \bigsqcup\{P(s',i) \mid s' \text{ a finite prefix of } s\} & \text{if } s \text{ infinite and } C_s \text{ empty} \end{cases}$$

It is easy to see that P is continuous, and that for infinite s with C_s not empty, P computes the unique function $f \in F$ indexed by s. We take M_D to be the determinate process that computes P.

Finally, we take M_O to be an oracle process that produces exactly the streams $s \in S$. Clearly, with this definition of M, M can behave like any and all the functions in F – that is, $\mathcal{F}_1[\![M]\!] = F$. Note that although M_D is defined for infinite streams not in S (and may not compute a function in F on such streams), restricting the oracle M_O to the set S assures that the "extra" functions are not possible behaviors for M.

∎

Theorem 4. The class of all networks constructible with infinity-fair merge is a proper subclass of the oraclizable networks.

Proof: By the following two lemmas.

Lemma 6. All networks constructible with infinity-fair merge and determinate processes are oraclizable.

Proof: Infinity-fair merge is oraclizable, since it is equivalent to an oracle that produces fair bit streams connected to a deterministic merge that uses the oracle input to decide which channel to read next. Since oraclizable networks are closed under composition, the result follows.

Lemma 7. The set of oraclizable networks has strictly greater cardinality than the set of infinity-fair merge networks.

Proof: As we have already noted, infinity-fair merge is equivalent to an oracle that produces all fair bit streams connected to a determinate merge. Hence any infinity-fair merge network can be implemented as this fair oracle connected to some determinate network. Thus, the number of infinity-fair merge networks is bounded by the number of determinate networks, which by Kahn's principle is the same as the number of continuous

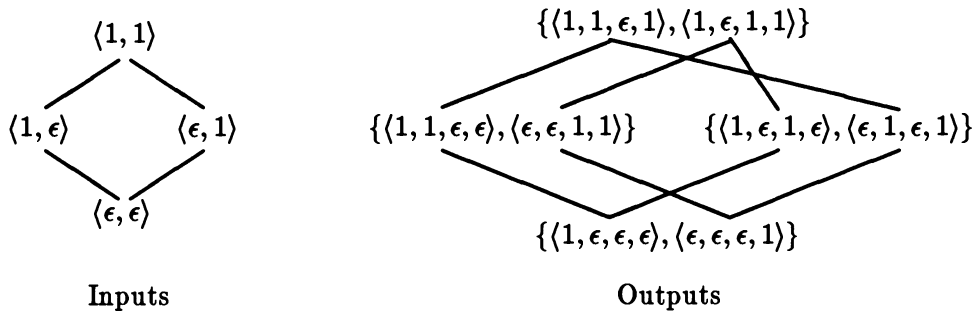


Figure 2: An Egli-Milner monotone input-output relation from S^2 to S^4.

stream-valued functions. Since the domain of streams is ω-algebraic, this is the same cardinality as the powerset of ω, $\mathcal{P}(\omega)$.

In general, the oracle part of an oraclizable network may emit *any* set of streams. Hence there are at least as many oraclizable networks as the powerset of the domain of streams, $\mathcal{P}(\mathsf{S})$. Since the domain of streams is as large as $\mathcal{P}(\omega)$, the cardinality of the set of oraclizable networks is at least that of $\mathcal{P}(\mathcal{P}(\omega))$, which is strictly greater than the cardinality of $\mathcal{P}(\omega)$.

∎

Finally, we show that the conjecture that the infinity-fair merge networks are exactly those whose input-output relations are Egli-Milner monotone fails to be true. In fact, we show that even the oraclizable networks fail to capture all of the Egli-Milner monotone input-output relations. Recall that given sets A and B, $A \sqsubseteq_{EM} B$ iff

$$\forall a \in A\ \exists b \in B \text{ s.t. } a \sqsubseteq b \ \ \& \ \ \forall b \in B\ \exists a \in A \text{ s.t. } a \sqsubseteq b.$$

We say that the input-output relation of a network M is Egli-Milner monotone iff $i \sqsubseteq i'$ implies

$$\{o | (i, o) \in \mathcal{IO}[\![M]\!]\} \sqsubseteq_{EM} \{o' | (i', o') \in \mathcal{IO}[\![M]\!]\}.$$

Theorem 5. The class of oraclizable networks (and hence the infinity-fair merge networks) is a proper subclass of the class of networks with Egli-Milner monotone input-output relations.

Proof. It is easily seen that all oraclizable networks have Egli-Milner monotone input-output relations. To see that the containment is proper, consider the input-output relation described in figure 2 (lines are drawn between comparable elements). It is clearly Egli-Milner monotone, but cannot be represented by any set of functions, since the "diamond" of relations among the inputs does not appear in the output. Since we know that sets of functions are universal for infinity-fair merge networks, no such network can implement this relation.

∎

6 Conclusions and Future Work

In the above we have considered the characterization and semantics of oraclizable networks in terms of sets of functions. Another possible representation of this class is on by certain functors relating appropriate categorical powerdomains. An approach along these lines may extend the results of this paper to broader classes of nondeterminism.

Another interesting direction is an investigation of the relations among classes of networks constructed using more powerful merge primitives, such as angelic merge or fair merge, and the oraclizable networks, Egli-Milner monotone networks, or networks with weaker monotonicity properties.

References

[Abr89] S. Abramsky. Unpublished lecture at MFPS, 1989.

[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.

[KP86] R. M. Keller and P. Panangaden. Semantics of digital networks containing indeterminate operators. *Distributed Computing*, 1(4):235–245, 1986.

[Mis89] J. Misra. Equational reasoning about nondeterministic processes. unpublished manuscript, 1989.

[MPS88] D. McAllester, P. Panangaden, and V. Shanbhogue. Nonexpressibilty 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.

[Rus89] J. R. Russell. Full abstraction for nondeterministic dataflow networks. To appear in Proceedings of the 30th Annual Symposium of Foundations of Computer Science, 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.