Denotational Semantics of Nets with Nondeterminism

Joost N. Kok *

We define a topological framework for streams of traces. With this approach Kahn's method generalizes to nets with bounded nondeterminism. We consider fixpoints of multivalued functions. We have a standard fixed point theorem, which can be used to model feed back loops. These fixed points can also be obtained by iteration. We give a general syntax of nets and see how we can analyze them in our streamframework. We show how to avoid the Brock-Ackerman and Keller anomalies. We are able to model the fair merge, which is a continuous function in our framework, and delay along lines. We prove a lemma that says that the order in which we connect nodes in our networks does not matter. If we have nets with nodes with unbounded nondeterminism, we can still use these fixpoints, but we do lose in our topological framework our iteration theorem.

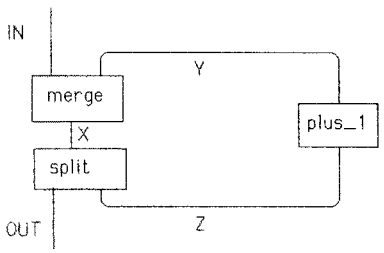
Note: this work has been carried out in the context of LPC: the dutch National Project for Concurrency, supported by the Netherlands Organisation for the Advancement of Pure Research (Z.W.O.), grant 125-20-04.

*
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

1. Introduction

Kahn [Kahn 1974] has introduced a semantic framework for deterministic dataflow nets. Many researchers have tried to extend these ideas to nets with nondeterministic nodes, for example [Keller 1978], [Brock & Ackerman 1981], [Arnold 1981], [Boussinot 1982]. See also the references listed at the end of this paper. A straightforward extension of Kahn's framework does not work, because serious anomalies arise, as is shown by [Keller 1978] and [Brock & Ackerman 1981]. We introduce these anomalies by two examples, for more details see the original papers.

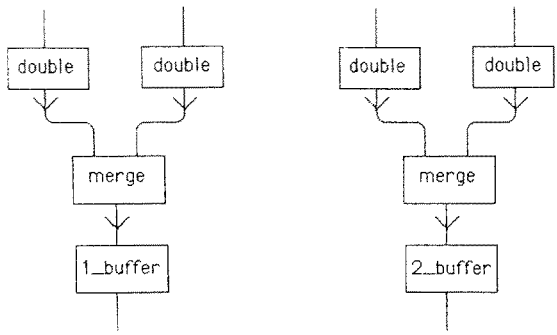
Example 1 (Keller Anomaly) Consider the following net:



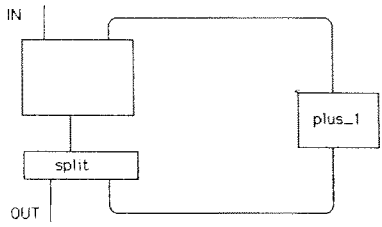
The tokens on this net will be integers. This net has one input- and one output line and it consists of three nodes: - a merge node which merges its two input lines, - a split node which outputs its input tokens on both output lines, - a plus_1 node which adds 1 to each token that passes this node. Now consider what happens if we put a "1" on the input line of this net. It passes the merge node, and arrives at the split node. The split node sends one copy of "1" to the output line and one copy to the plus_1 node, which adds 1 to "1" and sends "2" to the merge node. Continuing in this way we see that the desired output is the infinite stream 123... . Now let us look at what happens if we try to apply Kahn's method to this net. First write down the set of equations: {merge(IN , Y) = X, split(X) = < OUT , Z >, plus_1(Z) = Y}. The set of equations is derived from the net. IN, OUT, X, Y, Z represent histories on lines in our net. A history is a sequence of values. Let ϵ denote the empty history. The nodes are represented by functions that map histories to (sets of) histories. We are looking for a solution which can be obtained by iteration, as in Kahn's approach. We start by initialyzing X = Y = Z = OUT = ϵ and IN = 1. We now 'fire' the nodes and compute the 'new' values of these variables: 1. X=IN ={ 1 }, OUT=Z=Y={ ϵ }. We repeat this process (iteration): 2. X=IN=Z={ 1 }, Y={ ϵ }, 3. X=IN=OUT=Z={ 1 }, Y={ 2 }, 4. X={ 12, 21 }, IN=OUT=Z={ 1 }, Y={ 2 }, 5. X={ 12, 21 }, IN={ 1 }, OUT=Z={ 12, 21 }, Y={ 2 }. We can continue this way and get (in the limit) sets of histories on all the lines. Remark that now there can be some output (on OUT) that is not possible operationally. We see that in the limit there is an infinite stream which starts with 21. By operational intuition this is not allowed.

Example 2 (Brock-Ackerman anomaly). Consider the following two networks which each have

two input and two output lines:



-double is a node that when it receives a token will output it twice (for example double(12)=1122), -1_buffer is a buffer of length 1 that behaves like the identity function, -2_buffer is a buffer of length 2 that, if it contains two tokens outputs both, but if it contains only one token, it waits until it receives a second token. If we assume that all nodes work at a finite speed (not at zero speed) we see that networks "A" and "B" have the same input-output behaviour. The double nodes mask the difference between the two buffers. Now place the networks "A" and "B" in the following context:



If we insert subnet "A" and use "1" as input the output 12.. is possible, but if we insert subnet "B" this is not possible. Let us look more carefully at what happens. When we use subnet "A" and use as input token "1" this "1" will be doubled. Now imagine that one of these tokens remains for some time between the double node and the merge node. The other token passes through the merge node and the 1_buffer. After this it can go to the split and the plus_1 node. Now it comes back in the subnet "A" as "2" and can pass the merge node before the second "1" which was still at its previous position between the double node and the merge node. With subnet "B" this is not possible, because before we have some output from this net both "1" tokens must have passed the merge node. So we see, although nets "A" and "B" have the same input-output behaviour in isolation, they have different behaviour in some context. We will try to explain (informally) some of the concepts of this paper. We look at the behaviour of a node in a network. The behaviour consists of three stages: node consumes input, node works on this input, node produces output. When a node consumes input, it takes a sequence of tokens (called a *trace*) from each of its input lines. Then it works on this input, it does some internal processing and outputs traces on its output lines. After this output, the node starts again consuming input. Note that a node starts outputting after it has finished to consume tokens. The behaviour of a node can thus be described by a function that maps (tuples of) sequences of traces to (sets of

tuples of) sequences of traces. An infinite sequence of traces will be called a *stream.* A finite sequence of traces can be made into an infinite by adding ϵ's.

EXAMPLE: Consider a node that waits till it has received four integers and then outputs the sum of these four integers. The behaviour of such a node can be described by a function ϕ that maps streams to streams. Let ϵ denote the empty trace. Let $\Theta = <1111,1111, \cdots >$ and $\Theta' = <11,\epsilon,11,\epsilon, \cdots >$ be two streams. We have $\phi(\Theta) = <4,4, \cdots >$ and $\phi(\Theta') = <\epsilon,\epsilon,4,\epsilon,\epsilon,4, \cdots >$.

Now we are able to solve the anomalies: we can observe in our model things like "before we input some tokens there must be some output". For example $\phi(\ <\epsilon,34,\epsilon,\epsilon, \cdots >\ ) = <12,45,\epsilon,\epsilon, \cdots >$ can be translated as: first there was no input and the node produced the tokens 12, then the input was extended by 34 and this caused 45 to appear on the output line.

Now we have introduced the general idea, we give an overview of the rest of our paper. In section 2 we introduce some mathematical preliminaries about metric spaces. Section 3 describes our domains. We have a domain of tuples of streams and a domain of functions. Both are turned into metric spaces. Section 4 deals with the syntax of nets, and in section 5 we construct a mapping from these nets into our domains. In this section we prove a lemma about the order of connection of lines in our nets. Section 6 is about the generalization of functions and section 7 discusses the delay function. The next section indicates why the anomalies discussed before do not occur in our framework, and section 9 looks at what happens if we allow nodes with unbounded nondeterminism.

2. MATHEMATICAL PRELIMINARIES

In this section we collect some basic definitions and properties concerning metric spaces. Let X be any set. $\mathcal{P}(X)$ denotes the powerset of X, i.e., the set of all subsets of X. $\mathcal{P}_{\cdots}(X)$ denote the set of all subsets of X which have property $\cdots$. A sequence $x_0,x_1, \cdots$ of elements of X is usually denoted by $<x_i>_{i=0}^{\infty}$ or, briefly, $<x_i>_i$. For *limit*, *supremum* (sup), etc. of a sequence $<x_i>_i$. We use the notations $\lim_{i\to\infty} x_i$, or, briefly, $lim_i\ x_i$, $sup_i\ x_i$, etc.

DEFINITION 2.1. A metric space is a pair (M,d) with M a set and d (for *distance*) a mapping $d: M\times M\to[0,1]$ which satisfies the following properties: (a) $d(x,y)=0$ iff $x=y$, (b) $d(x,y)=d(y,x)$, (c) $d(x,y)\leqslant d(x,z)+d(z,y)$

DEFINITION 2.2. Let (M,d) be a metric space.

a. Let $<x_i>_i$ be a sequence in M. We say that $<x_i>_i$ *converges to* an element x in M called its limit, whenever we have: $\forall\ \epsilon > 0\ \exists\ N\ \forall\ n > N\ [\ d(x,x_n) < \epsilon\ ]$. A sequence $<x_i>_i$ in M is a convergent sequence if it converges to x for some $x \in X$

b. A sequence $<x_i>_i$ is called a *Cauchy sequence* whenever we have $\forall\ \epsilon > 0\ \ \exists\ N\ \forall\ n,m > N$

$[\, d(x_n, x_m) < \epsilon \,]$

c. The space (M,d) is called *complete* whenever each Cauchy sequence converges to an element in M.

d. A subset X of M is called *closed* whenever each Cauchy sequence in X converges to an element of X.

DEFINITION 2.3.

a. Let (M_1,d_1) and (M_2,d_2) be two metric spaces. We call the function $f : M_1 \to M_2$ *continuous*, whenever, for each sequence $<x_i>_i$ with limit x in M_1, we have that $lim_i\, f(x_i) = f(x)$.

b. Let (M,d) be a metric space and $f : M \to M$. We call f *contracting* if there exists a constant c, $0 \leqslant c < 1$, such that, for all $x,y \in M$, $d(\, f(x)\, ,\, f(y)\,) \leqslant c \,.\, d(\, x\, ,\, y\,)$.

c. Let (M,d) be a metric space and $f : M \to M$. We call f *non distance increasing* if for all $x,y \in M$, $d(\, f(x)\, ,\, f(y)\,) \leqslant d(\, x\, ,\, y\,)$.

PROPOSITION 2.4. Each contracting function is continuous.

For each metric space (M,d) we can define a metric $\hat{d}$ on the collection of its nonempty closed subsets, denoted by $\mathcal{P}_{nc}(\, M\,)$, as follows:

DEFINITION 2.5 (Hausdorff distance). Let (M,d) be a metric space, and let X, Y be nonempty subsets of M. We put $d'(x,Y) = inf_{y \in Y}\, d(x,y)$, and $\hat{d}(X,Y) = max(sup_{x \in X}\, d'(x,Y)\, ,\, sup_{y \in Y}\, d'(y,X))$.

PROPOSITION 2.6. Let (M,d) be a metric space and $\hat{d}$ as in def. 2.5.

a. $(\mathcal{P}_{nc}(M),\hat{d})$ is a metric space.

b. If (M,d) is complete then $(\mathcal{P}_{nc}(M),\hat{d})$ is complete. Moreover, for $<X_i>_i$ a Cauchy sequence in $(\mathcal{P}_{nc}(M),\hat{d})$ we have $lim_i\, X_i = \{\, lim_i\, x_i : x_i \in X_i,\ <x_i>_i$ *a Cauchy sequence in* $M\,\}$.

Proofs of proposition 2.6. can be found e.g. in [Dugundji 1966] or [Engelking 1977]. The proposition is due to Hahn [Hahn 1948]. Useful information on topologies on spaces of subsets can be found in [Michael 1951].

DEFINITION 2.7. Let A be an alphabet. We use A^* to denote the collection of all finite words over A and A^ω to denote the collection of all infinite words over A. We put $A^{st} =^{df} A^* \cup A^\omega$. We use ϵ for the empty word, a^* for the set of all finite sequences of a's, and a^ω for the infinite sequence of a's. Analogously we use notations such as $(ab)^*$ or $(ab)^\omega$, etc. We shall use $u,v,w, \cdots$ to range over A^{st} and $X, Y, \cdots$ for subsets of A^{st}.

DEFINITION 2.8.

a. For $u \in A^*$, $|\, u\,|$ denotes the *length* of u.

b. For $u,v \in A^{st}$ we put $u \leqslant v$ if there exists w such that $u.w = v$. We call u a *prefix* of v.
c. For $u,v \in A^{st}$, $u(n)$ denotes the prefix of u of length n, in case $|u| \geqslant n$. Otherwise, $u(n) = u$.

We turn A^{st} into a metric space by defining a distance $d : A^{st} \times A^{st} \rightarrow [0,1]$ as follows:
DEFINITION 2.9. For $u,v \in A^{st}$ we put $d(u,v) = 2^{-sup\{n \mid u(n)=v(n)\}}$ with the understanding that $2^{-\infty} = 0$.

PROPOSITION 2.10. (A^{st},d) and (A^{ω},d) are complete metric spaces.
The proof can be found, for example, in [Nivat 1979].

3. DOMAINS

Let $\mathbb{N}$ be the set of integers. Take as alphabet $A = \mathbb{N}^{st}$. Define $DOM = A^{\omega}$. Members of DOM will be surrounded by $<$ and $>$ to avoid confusion with members of $\mathbb{N}^{st}$. In the sequel we will call members of DOM *streams* and members of $\mathbb{N}^{st}$ *traces*. The empty trace will be denoted by ϵ. If we give DOM the metric d of definition 2.9 we get by proposition 2.10 a complete metric space. For example $d(<123,456,7,1,1, \cdots >, <123,457,6,2,2, \cdots >) = 1/4$. Nodes in general have more than one input line, so we consider tuples of streams. To be more formal let $\Theta \in DOM^n$, where DOM^n is the set of n-tuples of streams. Let $\bar{d}: DOM^n \times DOM^n \rightarrow [0,1]$ be defined by $\bar{d}(\Theta,\bar{\Theta}) =^{df} \max_{i \in \{1,\ldots,n\}} d(\Theta_i,\bar{\Theta}_i)$. Usually we omit the bar of $\bar{d}$.

PROPOSITION 3.1. (DOM^n,d) is a complete metric space for each n.

By proposition 2.6 we have
COROLLARY 3.2. $(\mathcal{P}_{nc}(DOM^n),\hat{d})$ is a complete metric space for each n.

Let $\phi \in DOM^n \rightarrow \mathcal{P}_{nc}(DOM^m)$. Now define $DOM^{n:m}$ as follows: $DOM^{n:m} = \{\phi \mid \phi$ is non distance increasing $\}$ Non distance increasing is the generalization of monotonicity in order-theoretic frameworks. It says something like: if two inputs differ after n timesteps, the outputs will differ after n timesteps or later. If $\phi \in DOM^n \rightarrow \mathcal{P}(DOM^n)$ we can define the set of fixpoints: $FP(\phi) = \{\Theta \mid \Theta \in \phi(\Theta)\}$. In topology this is a standard way of defining fixed points of multivalued functions, see [Nadler 1970]. It is different from certain approaches in semantics: for example [de Bakker & Kok 1985] first generalize a function $\phi \in DOM^n \rightarrow \mathcal{P}_{nc}(DOM^n)$ to a function $\hat{\phi} \in \mathcal{P}_{nc}(DOM^n) \rightarrow \mathcal{P}_{nc}(DOM^n)$ and then solve the equation $\phi(X) = X$. Define $\mathbb{D} = \bigcup_n DOM^n \cup \bigcup_{n,m} DOM^{n:m}$. $\mathbb{D}$ will be our semantic domain. We have the following facts:

THEOREM 3.3. Let $\phi \in DOM^n \rightarrow \mathcal{P}_{nc}(DOM^n)$ be a continuous function. Then we have
i. $FP(\phi) \in \mathcal{P}_c(DOM^n)$
ii. $FP(\phi) = \{\lim_i \Theta^i \mid \Theta^0$ arbitrary, $\Theta^{i+1} \in \phi(\Theta^i), <\Theta^i>_i$ a Cauchy sequence $\}$
PROOF: i. Let $<\Theta^i>_i$ be a Cauchy sequence in $FP(\phi)$. Let $\Theta = lim_i \Theta^i$. We have by 2.3.a

$\phi(\Theta)=\lim_i \phi(\Theta^i)$. For each i $\Theta^i \in FP(\phi)$, so $\Theta^i \in \phi(\Theta^i)$, so by 2.6.b we have $lim_i \Theta^i \in \phi(\Theta)$, so $\Theta \in \phi(\Theta)$ thus $\Theta \in FP(\phi)$.

ii. Let $<\Theta^i>_i$ be a Cauchy sequence such that for each i $\Theta^{i+1} \in \phi(\Theta^i)$. We have $\phi(lim_i \Theta^i) = lim_i \phi(\Theta^i) \ni lim_i \Theta^{i+1} = lim_i \Theta^i$ so $lim_i \Theta^i \in FP(\phi)$.

The other way is easy: if $\Theta \in FP(\phi)$, then we have $\Theta \in \phi(\Theta)$, so $<\Theta, \Theta, \cdots>$ is the desired Cauchy sequence.□

THEOREM 3.4. Let $\phi \in DOM^n \to \mathcal{P}_{nc}(DOM^n)$ be a contracting function. Then we have that $FP(\phi)$ is non empty.

PROOF Cauchy sequences of theorem 3.3 are easy to construct: take a Θ^0 arbitrary, choose $\Theta^1 \in \phi(\Theta^0)$. Let $d(\Theta^0, \Theta^1) = \alpha$. By the contractivity of ϕ we have $\hat{d}(\phi(\Theta^0), \phi(\Theta^1)) \leqslant c.\alpha$, $0 \leqslant c < 1$, so there exists a $\Theta^2 \in \phi(\Theta^1)$ such that $d(\Theta^1, \Theta^2) \leqslant c.\alpha$. Continuing this way, we get members of $FP(\phi)$. □.

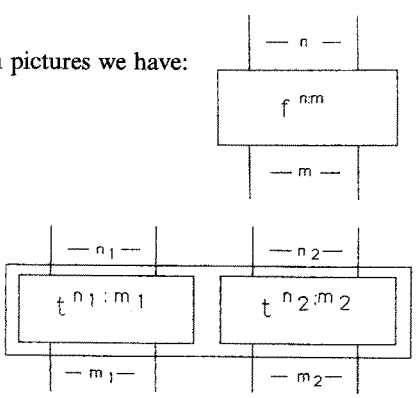
The proof of theorem 3.4 is given, in a more general setting in [Nadler 1970].

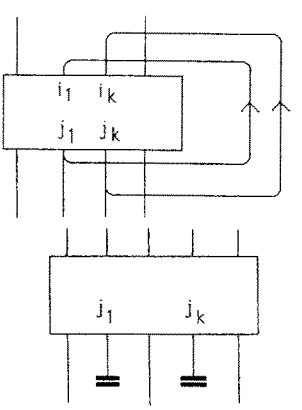
4. SYNTAX OF NETS

A net is described by a term, which has an arity, and this arity corresponds to the numbers of the input- and the output lines. First we define the class of standard nodes: (for example *merge* $\in Node^{2:1}$) $d^{n:m} \in Node^{n:m}$, $d \in Node = \bigcup_{n,m} Node^{n:m}$. Let net expressions be defined by: $t \in NetExp = \bigcup_{n,m} NetExp^{n:m}$ where $NetExp^{n:m}$ can be build up in the following way:

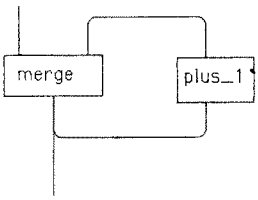
$t^{n:m}$	$::= d^{n:m}$	(standard node)
	$\mid < t_1^{n_1:m_1}, t_2^{n_2:m_2} >$	(disjoint union, $n_1+n_2=n, m_1+m_2=m$)
	$\mid t^{n+k:m}\{i_1:j_1, \cdots, i_k:j_k\}$	(feedback, i_l input is connected to j_l output, $1 \leqslant i_1 < \cdots < i_k \leqslant n+k, \forall\, l, 1 \leqslant l \leqslant k,\ 1 \leqslant j_l \leqslant m$)
	$\mid t^{n:m+k} / \{j_1, \cdots, j_k\}$	(abstraction: we are not interested in the $j_1 \cdots j_k$ output lines, $1 \leqslant j_1 < \cdots < j_k \leqslant m+k$)

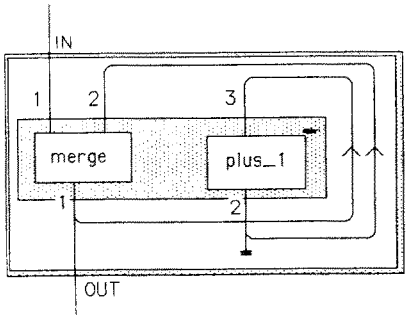
In pictures we have:





Example: the following picture is represented in our syntax as (< *merge* , *plus*_1 > { 2:2 , 3:1 }) / { 2 }





Compare with example 1.

5. SEMANTICS

First we need some definitions. Let $\Theta \in DOM^n$ and $X \in \mathcal{P}(DOM^n)$.

(*Projection*) $X_{<i_1,\cdots,i_k>} = \bigcup_{\Theta \in X} \Theta_{<i_1,\cdots,i_k>}$ and $\Theta_{<i_1,\cdots,i_k>} = <\Theta_{i_1},\cdots,\Theta_{i_k}>$.

(*Slicing*) $X_{(i_1,\cdots,i_k)} = \bigcup_{\Theta \in X} \Theta_{(i_1,\cdots,i_k)}$ and $\Theta_{(i_1,\cdots,i_k)} =$ "Θ *without* $\Theta_{i_1},\cdots,\Theta_{i_k}$"

$< \Theta_1, \cdots, \Theta_{i_1-1}, \Theta_{i_1+1}, \cdots, \Theta_{i_k-1}, \Theta_{i_k+1}, \cdots, \Theta_n >$

(Concatenation) $\circ : \mathbb{N}^{st} \times DOM^n \to DOM^n$ $<x_1,\ldots,x_n> \circ X = \bigcup_{\Theta' \in X} <x_1,\ldots,x_n> \circ \Theta$

and $<x_1,\ldots,x_n> \circ \Theta = < x_1 \circ \Theta_1, \cdots, x_n \circ \Theta_n >$

(*Combination of two functions*) Let $\phi_1 \in DOM^{n_1:m_1}$ and $\phi_2 \in DOM^{n_2:m_2}$.

$<\phi_1,\phi_2> \in DOM^{n_1+n_2:m_1+m_2}$

$<\phi_1,\phi_2>(\Theta) = \{\overline{\Theta} \in DOM^{m_1+m_2} \mid \overline{\Theta}_{<1,\ldots,m_1>} \in \phi(\Theta_{<1,\ldots,n_1>})$

and $\overline{\Theta}_{<m_1+1,\ldots,m_1+m_2>} \in \phi(\Theta_{<n_1+1,\ldots,n_1+n_2>})\}$

(*Combination of two tuples*) For each $k,l \in \mathbb{N}$ we define a $Join_{k,l}$ function:

$$Join_{k,l} : \{<i_1,\ldots,i_k> \mid 0 \leq i_1 < \cdots < i_k \leq k+l\} \times DOM^k \times DOM^l \to DOM^{k+l}$$

$$Join_{k,l}(<i_1,\ldots,i_k>,\Theta,\overline{\Theta}) = \Theta' \text{ where } \Theta'_{<i_1\cdots i_k>} = \Theta\Theta'_{(i_1\cdots i_k)} = \overline{\Theta}$$

Usually we omit the subscripts k,l of the *Join* function.

Examples: let $\Theta \in DOM^3$ be defined by $\Theta = <<1,1,\cdots>,<2,2,\cdots>,<3,3,\cdots>>$ Then $\Theta_{<1,2>} = <<1,1,\cdots>,<2,2,\cdots>>$ and $\Theta_{(1,2)} = <<3,3,\cdots>>$. We also have $<12,34,56> \circ \Theta = <<12,1,1,\cdots>,<34,2,2,\cdots>,<56,3,3,\cdots>>$. Let $\Theta' \in DOM^2$ be defined by $\Theta' = <<4,4,\cdots>,<5,5,\cdots>>$. Then $Join(<1,3,5>,\Theta,\Theta') = Join(<2,4>,\Theta',\Theta) = <<1,1,\cdots>,<4,4,\cdots>,<2,2,\cdots>,<5,5,\cdots>,<3,3,\cdots>>$.

Let *Env* be the set of environments, $\gamma \in Env$, $Env = Node \to DOM$. Each environment has to be

type preserving, i.e. if $d \in Node^{n:m}$ then we must have $\gamma(d) \in DOM^{n:m}$. Now we are sufficiently prepared to give the semantics of net expressions. Let $[\![.]\!] : NetExp \to Env \to \mathbb{D}$ be defined as follows:

$$[\![d^{n:m}]\!]\gamma = \gamma(d^{n:m})$$

$$[\![< t_1^{n_1:m_1}, t_2^{n_2:m_2} >]\!]\gamma = < [\![t^{n_1:m_1}]\!]\gamma , [\![t^{n_2:m_2}]\!]\gamma >$$

$$[\![t^{n:m+k} / \{ j_1, \cdots, j_k \}]\!]\gamma = \lambda\Theta.(([\![t^{n:m+k}]\!]\gamma)(\Theta))_{(j_1 \cdots j_k)}$$

$$[\![t^{n+k:m}\{i_1{:}j_1, \ldots, i_k{:}j_k\}]\!]\gamma = \lambda\Theta.FP(\lambda\overline{\Theta}.\phi(Join(<i_1, \ldots, i_k>, <\epsilon, \ldots, \epsilon> \circ \overline{\Theta}_{<j_1, \ldots, j_k>}, \Theta)))$$

where $\phi = [\![t^{n+k:m}]\!]\gamma$. Observe, that if $\phi \in DOM^{n+k:m}$, then for all Θ, $\lambda\overline{\Theta}.\phi(Join(<i_1, \ldots, i_k>, <\epsilon, \ldots, \epsilon> \circ \overline{\Theta}_{<j_1, \ldots, j_k>}, \Theta)))$ is contractive, so we can apply 3.3.ii. The theorem and generalizations of it say that it does not matter in which order we connect the input and output lines.

THEOREM 5.1. For all $\gamma \in Env$, for all $t^{2:2} \in NetExp^{2:2}$

$$[\![t^{2:2}\{1{:}1, 2{:}2\}]\!]\gamma = [\![(t^{2:2}\{1{:}1\})\{1{:}2\}]\!]\gamma$$

PROOF. Let $\phi = [\![t^{2:2}]\!]\gamma$. We have $\phi \in DOM^{2:2}$. $[\![t^{2:2}\{1{:}1, 2{:}2\}]\!]\gamma = \{\Theta \in DOM^2 \mid \Theta \in \phi(\epsilon \circ \Theta_1, \epsilon \circ \Theta_2)\}$. Recall that Θ_1 denotes projection on the first coördinate. $[\![t^{2:2}\{1{:}1\}]\!]\gamma = \lambda\Theta'.\{\Theta'' \in DOM^2 : \Theta'' \in \phi(\epsilon \circ \Theta''_1, \Theta')\} =^{df} \phi'$. We have $\phi' \in DOM^{1:2}$. $[\![(t^{2:2}\{1{:}1\})\{1{:}2\}]\!]\gamma = \{\Theta''' \in DOM^2 : \Theta''' \in \phi'(\epsilon \circ \Theta'''_2)\} = \phi''$. We have: $\phi'' = \{\Theta''' \in DOM^2 : \Theta''' \in \{\Theta'' \in DOM^2 \mid \Theta'' \in \phi'' \in \phi(\epsilon \circ \Theta''_1\ \epsilon \circ \Theta'''_2)\}\} = \{\Theta''' \in DOM^2 : \Theta''' \in \phi(<\epsilon \circ \Theta'''_1, \epsilon \circ \Theta'''_2>)\}$. □

6. GENERALIZATION OF FUNCTIONS

Usually functions in networks are not an element of $DOM^{n:m}$. Let there be given a function $\phi : (\mathbb{N}^{st})^n \to \mathcal{P}((\mathbb{N}^{st})^m)$, We extend this function ϕ to a function $\hat{\phi} \in DOM^{n:m}$. Take a certain input $\Theta = <<\Theta_{11}, \Theta_{12}, \cdots>, \ldots, <\Theta_{n1}, \Theta_{n2}, \cdots>>$. We define

$$\begin{aligned}
\hat{\phi}(\Theta) = \{\, \overline{\Theta} \mid\ & (<\overline{\Theta}_{11}, \ldots, \overline{\Theta}_{m1}> \in \phi(<\Theta_{11}, \ldots, \Theta_{n1}>) \\
& \vee (<\overline{\Theta}_{11}, \ldots, \overline{\Theta}_{m1}> = <\epsilon, \ldots, \epsilon> \wedge \phi(<\Theta_{11}, \ldots, \Theta_{n1}>) = \varnothing)) \\
& \wedge \\
& (<\overline{\Theta}_{11}\overline{\Theta}_{12}, \ldots, \overline{\Theta}_{m1}\overline{\Theta}_{m2}> \in \phi(<\Theta_{11}\Theta_{12}, \ldots, \Theta_{n1}\Theta_{n2}>) \\
& \vee (\phi(\overline{\Theta}_{11}\overline{\Theta}_{12}, \ldots, \overline{\Theta}_{m1}\overline{\Theta}_{m2}) = \varnothing \wedge <\overline{\Theta}_{12}, \ldots, \overline{\Theta}_{n2}> = <\epsilon, \ldots, \epsilon>)) \\
& \wedge \cdots \}
\end{aligned}$$

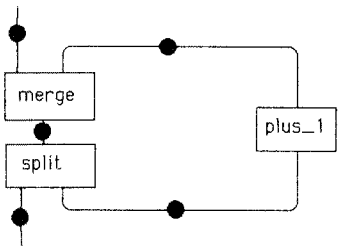
We give an example. The generalized (fair) merge function behaves as follows: $merge(<<12,\epsilon,\epsilon, \cdots>, <34,\epsilon,\epsilon, \cdots>>) = \{<1234,\epsilon,\epsilon, \cdots>, <1324,\epsilon,\epsilon, \cdots>, <1342,\epsilon,\epsilon, \cdots>, <3124,\epsilon,\epsilon, \cdots>, <3412,\epsilon,\epsilon, \cdots>, <3142,\epsilon,\epsilon, \cdots>\}$ and

merge(<<1,2,ϵ,ϵ, $\cdots$ >, <3,4,ϵ,ϵ, $\cdots$ >>)= {<13,24,ϵ,ϵ, $\cdots$ >, <31,24,ϵ,ϵ, $\cdots$ >, <13,42,ϵ,ϵ, $\cdots$ >, <31,42,ϵ,ϵ, $\cdots$ >}

Remark: it is possible that this construction does not give the desired results if we generalize functions with unbounded nondeterminism. For a discussion see section 9.

7. Delay function

Our model can also be used to model delay along lines, by the introduction of delay functions. On each line of the network we place a delay node. For example the network of example 1 is replaced by:



$\hat{delay} = \hat{PREF}$ where $PREF : \mathbb{N}^{st} \to \mathcal{P}(\mathbb{N}^{st})$ is defined by $PREF(x) = \{ x' \mid x' \leqslant x \}$ so the delay function is the generalization of the prefix function.

8. The absence of the Brock-Ackerman and Keller anomalies

We show why these anomalies do not occur in our framework. *Keller anomaly:* we use the idea of extension [de Bakker et al 1985]. It is somewhat hidden in the definition of the generalization function "$\hat{}$" of section 5. In modeling a feed back loop we insist that if we accept a new tuple as output, it must extend previous outputs. Therefore it is not possible for new tokens to pass old ones. *Brock-Ackerman anomaly:* with our extended streams it is possible to use some abstract time, although not so explicit as was suggested by Wadge as in [Wadge 1981], where τ's are introduced to represent ticks of a clock. Our semantics gives a different meaning to the networks "A" and "B" of section 1. For example with input <1,ϵ,2,ϵ,ϵ, $\cdots$ > output <ϵ,1,2,ϵ,12,ϵ, $\cdots$ > is possible for "A" but not for "B".

9. Nodes with unbounded nondeterminism

In the previous sections we used as domains for functions $DOM^n \to \mathcal{P}_{closed}(DOM^m)$. We furthermore required that these functions are non distance increasing. In this section we look what happens if we allow non closed sets, i.e. take as domains $DOM^n \to \mathcal{P}(DOM^m)$. Functions that are not an element of a $DOM^n \to \mathcal{P}_{closed}(DOM^m)$ but are element of a $DOM^n \to \mathcal{P}(DOM^m)$ for certain n,m, are said to be functions with *unbounded* nondeterminism.

First of all, we can not use the Hausdorff metric for $\mathcal{P}(DOM^m)$, because it is only a metric on $\mathcal{P}_{closed}(DOM^m)$. As a consequence, we do not have our iteration theorem 3.3 at hand. We must

reformulate the 'non distance increasing' property. But we can still define $FP(\phi)=\{\Theta:\Theta\in\phi(\Theta)\}$, but this fixed point can (in general) not be obtained by iteration.

We can define our map $[\![.]\!]$: $NetExp\rightarrow Env'\rightarrow\mathbb{D}'$ as in section 5, where $Env'=Node\rightarrow\mathbb{D}'$ and $\mathbb{D}'=$

$$\bigcup_n DOM^n \cup \bigcup_{n,m}\{\phi\in DOM^n\rightarrow\mathcal{P}(DOM^m):\forall\Theta,\Theta'\forall k.\Theta[k]=\Theta'[k] \text{ implies } \phi(\Theta)[k]=\phi(\Theta')[k]\}$$

where $\cdots$[k] denotes truncation of length k. Now we consider the generalization function "$\hat{}$" of section 5. $\hat{}$: $((\mathbb{N}^{st})^n\rightarrow\mathcal{P}((\mathbb{N}^{st})^m)\rightarrow DOM^{n:m}$. If we have a function member of $((\mathbb{N}^{st})^n\rightarrow\mathcal{P}((\mathbb{N}^{st})^m)$, but not a member of $((\mathbb{N}^{st})^n\rightarrow\mathcal{P}_{closed}((\mathbb{N}^{st})^m)$, this construction does not always yield the right result. The operationally desired function is then not a member of $DOM^{n:m}$, but is a member of $DOM^n\rightarrow\mathcal{P}(DOM^m)$. We have to use another generalization function. We therefore must be able to record which output lines depend on which input lines. We can do this with coloring of the input lines. [de Bakker et al 1985] use a coloring of bottom elements, but for a different purpose.

10. Comparison with other frameworks

Broy [Broy 1984] and Park [Park 1983] work with oracles. With oracles, a merge node can be made deterministic and fair if we use only fair oracles. Indeed, the Brock-Ackerman anomaly can be solved this way, but in our opinion the use of oracles is not the most elegant solution. An algebraic approach is also possible, see for example [Back & Mannila 1982] or [Bergstra & Klop 1982]. Dataflow is modeled by processes, which can communicate by sending each other tokens. Delay is modeled by buffers. There are, however, some restrictions on the functions that can be used in these frameworks. Order theoretic approaches used up to now have either difficulties with defining an ordering on sets (see for example the discussion in [Abramsky 1983]), or there are no real fixed points, for example in [de Bakker et al 1985] or [Staples & Nguyen 1985]. We do not need all the information that is available in the scenarios of Brock and Ackerman [Brock & Ackerman 1981]. The same applies to a lot of category theoretic models, in which all information is available. Most of the topological frameworks work in a metric setting with closed or compact sets. Our closedness is less "serious". For example 1^* is not closed (it does not contain 1^ω), but $\{<1,\epsilon,\cdots>,<11,\epsilon,\cdots>,<111,\epsilon,\cdots>,...\}$ is closed, although it does not contain $<1^\omega,\epsilon,\cdots>$. With unbounded nondeterminism we can still apply our definitions, but we do not have our nice metric topological framework at hand. Future work will try to integrate our framework with that of Keller and Panangaden [Keller & Panangaden 1985], combining our fixed point technique with their categorical approach via event structures.

11. REFERENCES

[Abramsky 1983], A. Abramsky, *On Semantic Foundations for Applicative Multiprogramming,* Proc. 10th ICALP, (J.Diaz ed.), Barcelona, LNCS 154, Springer, 1983, pp. 1-14.

[Arnold 1981], A. Arnold, *Semantique des Processus Communicants,* RAIRO 15 (2), 1981, pp.103-109.

[Back & Mannila 1982], R.J.Back, N. Mannila, *A Refinement of Kahn's Semantics to Handle Nondeterminism and Communication,* Proc. ACM Symp. on Distributed Comp., Ottawa, 1982, pp. 111-120.

[de Bakker et al 1985], J.W. de Bakker, J.-J.Ch. Meyer, J. Zucker, *Bringing Color in the Semantics of Nondeterministic Dataflow,* Preprint, Centre fo Mathematics and Computer Science, 1985.

[de Bakker & Kok 1985], J.W. de Bakker, J.N. Kok, *Towards a Topological Treatment of Streams and Functions on Streams,* Proc. 12th ICALP, (W. Brauer ed.), Nafplion, LNCS 194, 1985, pp. 140-149.

[Bergstra & Klop 1983], J. Bergstra, J.W. Klop, *Process Algebra for the Operational Semantics of Static Dataflow Networks,* Techn. Report Mathematical Centre IW 222/83, Amsterdam, 1983.

[Boussinot 1982], Boussinot, *Proposition de semantique denotationelle pour des reseaux de processus avec operateur de melange equitable,* TCS 18, 1982, pp. 173-206.

[Brock & Ackerman 1981], J.D. Brock, W.B. Ackerman, *Scenarios: A Model of Non-determinate Computation,* in Proc. Formalization of Language Concepts, (J. Diaz, J. Ramos eds.), LNCS 107, Springer, 1981, pp. 252-259.

[Broy 1983], M. Broy, *Fixed Point Theory for Communication and Concurrency,* in: Formal Description of Programming Concepts-II, (Bjorner ed.), North-Holland, Amsterdam, 1983, pp. 125-148.

[Broy 1984], M. Broy, *Nondeterministic Data Flow Programs: How to avoid the Merge Anomaly,* preprint, Fakultat fur Mathematik und Informatik, Universitat Passau, 1984.

[Dugundji 1966], J. Dugundji, *Topology,* Allen and Bacon Rockleigh, N.J. 1966.

[Engelking 1977], R. Engelking, *General topology,* Polish Scientific Publishers 1977.

[Faustini 1982], A.A. Faustini, *An Operational Semantics for Pure Dataflow,* in: Proc. 9th ICALP, (M. Nielsen , E.M. Schmidt, eds.), LNCS 140, Springer, 1982, pp. 212-224.

[Hahn 1948], H. Hahn, *Reelle Funktionen,* Chelsea, New York, 1948.

[Kahn 1974] G. Kahn, *The Semantics of a Simple Language for Parallel Programming,* in: Proc. IFIP74, North-Holland, Amsterdam, 1977, pp. 993-998.

[Kahn & MacQueen 1977], G. Kahn, D.B. MacQueen, *Coroutines and Networks of Parallel Processes,* in Proc. IFIP 1977, North-Holland, Amsterdam, 1977, pp. 993-998.

[Keller 1978], R.M. Keller, *Denotational Models for Parallel Programs with Indeterminate Operators,* in: Formal Description of Programming Concepts, (E.J. Neuhold ed.), North-Holland, Amsterdam, 1977, pp. 337-366.

[Keller & Panangaden 1985], R.M. Keller, P. Panangaden, *Semantics of Networks Containing Indeterminate Operators,* in: Seminar on Concurrency, Carnegie-Mellon University, (S.D. Brookes, A.W. Roscoe, G. Winskel eds.), Lecture Notes in Computer Science 197, pp. 479-496, 1985.

[Kosinski 1978], P.R. Kosinski, *A Straightforward Denotational Semantics for Nondeterminate Data*

Flow Programs, in: 5th ACM POPL, 1978, pp. 214-221.

[Michael 1951], E. Michael, *Topologies on spaces of subsets,* Trans. AMS 71 (1951), pp12-182.

[Nadler 1970], Nadler, S.B., *Some Results on Multi-Valued Contraction Mappings,* in Set-Valued Mappings, Selections and Topological Properties of 2^X, (W.M. Fleischman ed), Lecture Notes in Mathematics, pp. 64-69, 1970.

[Nivat 1979], M. Nivat, *Infinite words, infinite trees, infinite computations,* Foundations of Computer Science III. 2, Mathematical Centre Tracts 109 (1979) 3-52.

[Park 1983], D. Park, *The Fairness Problem and Nondeterministic Computing Networks,* in: Foundations of Computer Science IV.2, (J.W. de Bakker, J. van Leeuwen eds), Mathematical Centre Tracts 159, Amsterdam, 1983, pp. 133-161.

[Staples & Nguyen 1985], J. Staples, V.L. Nguyen, *A Fixpoint Semantics for Nondeterministic Dataflow,* Journal of the ACM, april 1985, 32(2), 1985, pp. 411-445.

[Wadge 1981], W.W. Wadge, *An extensional Treatment of Dataflow Deadlock,* in Theoretical Computer Science 13 (1981), pp. 3-15, 1981.