**Centrum voor Wiskunde en Informatica**
Centre for Mathematics and Computer Science

J.N. Kok

A fully abstract semantics for data flow nets

Computer Science/Department of Software Technology · Report CS-R8724 May

The Centre for Mathematics and Computer Science is a research institute of the Stichting Mathematisch Centrum, which was founded on February 11, 1946, as a nonprofit institution aiming at the promotion of mathematics, computer science, and their applications. It is sponsored by the Dutch Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O.).

# A Fully Abstract Semantics for Data Flow Nets

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

Two semantic models for data flow nets are given.
The first model is an intuitive, operational model. This model has an important drawback: it is not compositional. An example given in [Brock & Ackerman 1981] shows the non-compositionality of our model. There exist two nets that have the same semantics, but when they are placed in a specific context, the semantics of the resulting nets differ.
The second one is obtained by adding information to the first model. The amount of information is enough to make it compositional. Moreover, we show that we have added the minimal amount of information to make the model compositional: the second model is fully abstract with respect to the equivalence generated by the first model.
To be more specific: the first model describes the semantics a data flow net as a function from (tuples of) sequences of tokens to sets of (tuples of) sequences of tokens. The second one maps a data flow net to a function from (tuples of) infinite sequences of finite words to sets of (tuples of) infinite sequences of finite words.

*1980 Mathematics Subject Classification:* 68B10.
*1982 CR Categories:* D.3.1, F.3.2, F.3.3.
*Key Words & Phrases:* dataflow programming, dataflow networks, denotational semantics, multivalued functions, concurrency.

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

*Note:* This paper will also appear in the proceedings of the PARLE conference. These proceedings will be published in the Lecture Notes on Computer Science (LNCS) series of Springer Verlag.

## 1. Introduction

In 1974 Kahn [Kahn 1974] gave a semantics for a certain class of data flow nets. His model is based on histories: he uses sequences of tokens. The nodes have to behave deterministically and should be continuous with respect to the prefix ordering on the sequences of tokens. Ever since that time researchers have tried to extend his ideas to more general classes of networks. Examples can be found in [Keller 1978], [Brock & Ackerman 1981], [Arnold 1981], [Boussinot 1982], [Park 1983], [Staples & Nguyen 1985]. A straightforward extension of Kahn's framework does not work for all kinds of nodes and networks.
One of the problems was first shown in [Brock & Ackerman 1981]. They showed that a semantics where a network is modeled as a function from tuples of words of tokens to sets of tuples of words of tokens, is not sufficient to obtain a compositional model. (We use sequences and words synonymously.) They give two networks $t_1$ and $t_2$ that have the same semantics: $[\![t_1]\!]=[\![t_2]\!]$. They construct a context $C()$ such that $[\![C(t_1)]\!]\neq[\![C(t_2)]\!]$. We give this example. Consider the two nets of figure 1.1.
We have three kinds of nodes in the nets $t_1$ and $t_2$. The node *dup* is a node which duplicates each token it receives, and sends both to the output line, *merge* is a node which merges its two inputs and *2 buffer* is a node which waits for a second token if it has received one, and then outputs both tokens. It is clear that any semantics which describes these two networks as a function from a tuple of sequences of tokens to a set of sequences of tokens should assign the same meaning to $t_1$ and $t_2$. The difference between the two nets is masked by the duplicate nodes. Consider the context of figure 1.2. When we place $t_1$ and $t_2$ in this context, the resulting nets have a different semantics. If we plug the nets in the context, we connect one of the input lines and the output line. Hence we obtain a net with one input and one output. The resulting nets have a different semantics. For example, in $[\![C(t_1)]\!](1)$ we have a sequence which starts with 12. In $[\![C(t_2)]\!](1)$ all sequences start with 11. In $t_1$ we can have

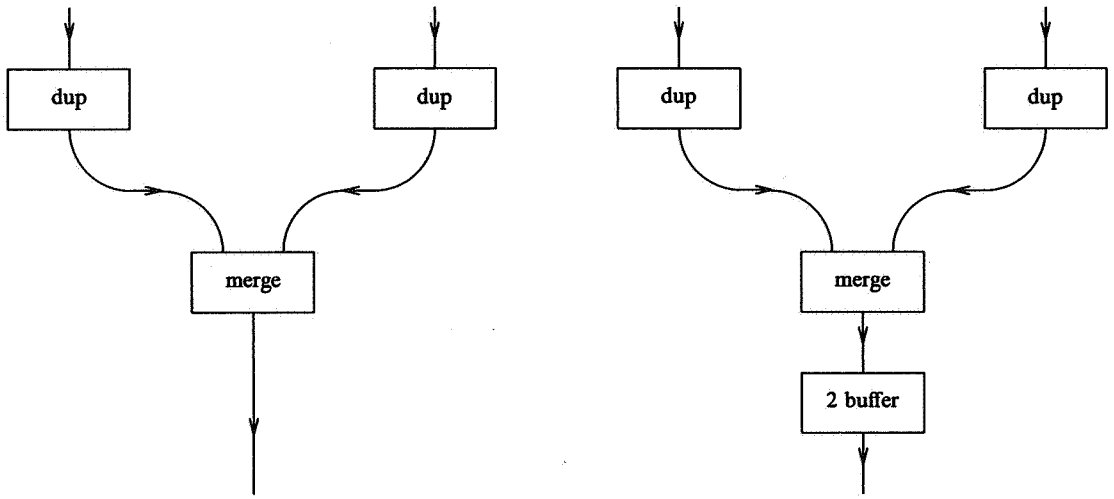


FIGURE 1.1

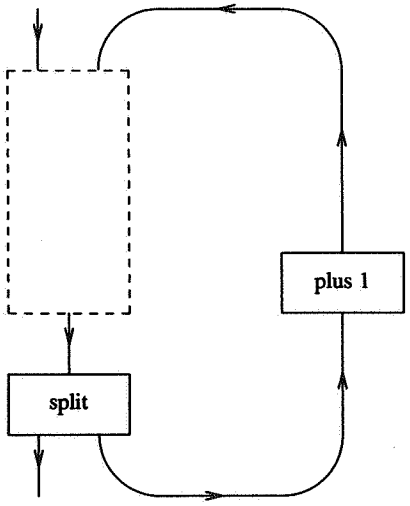


FIGURE 1.2

that a token can spend some time between the *dup* node and the *merge* node. A second token can go around the feed back loop and pass before the token that is waiting between the two nodes. In $t_2$ this is impossible: the *2 buffer* pulls the second token down.

The semantics is not fine enough: it can not make all the distinctions necessary. A solution is to add some (time) information to the words. The scenarios as proposed by Brock & Ackerman are an example of this addition of information.

The main difficulty with adding information is that we can add too much information. A criterion to

test if we have the right amount of information is to see whether or not the semantics is fully abstract. We define an equivalence relation $\equiv_0$ with the help of an operational semantics. The equivalence relation $\equiv_0$ on data flow nets is defined by: $t_1 \equiv_0 t_2$ iff $\mathcal{O}[\![ t_1 ]\!] = \mathcal{O}[\![ t_2 ]\!]$. Two nets are related by $\equiv_0$ iff they have the same input/output behavior in terms of words. A semantics $\mathcal{D}$ for data flow nets is called fully abstract with respect to this equivalence relation $\equiv_0$ if for all nets $t_1$ and $t_2$ we have $\mathcal{D}[\![ t_1 ]\!] = \mathcal{D}[\![ t_2 ]\!]$ iff $C(t_1) \equiv_0 C(t_2)$ holds for every context $C()$.
In this paper we give a denotational semantics that is fully abstract with respect to $\equiv_0$. It models a net by giving a function that maps a (tuple of) infinite sequences of finite words to a set of (tuples of) infinite sequences of finite words.
The outline of the rest of this paper is as follows: in section 2 we give the operational semantics, in section 3 the denotational semantics, section 4 relates the two, section 5 gives the fully abstractness result, and we conclude with section 6 where we compare our framework with other frameworks.

## 2. Operational Semantics

Assume that there is given a set of nodes *Node*. Let $d$ be a typical member of this set of nodes. The set $Node^{n:m}$ is the set of nodes with $n$ inputs and $m$ outputs. We construct an automaton for every node $d \in Node$. This automaton can write and read words from tapes. Such an automaton has, in general, an internal state. Let $S$ be the set of states and let $s$ be a typical element. The automaton starts working in an initial state. The behavior of a node $d \in Node^{n:m}$ is specified by a function

$$\delta : (A^*)^n \times S \to P(S \times (A^*)^m)$$

The $P$ denotes "subsets of". Note that the empty set is allowed. This function is called its specification. We assume for each node that such a specification is given. The intuitive meaning of a node

$$(\tilde{s},(\tilde{x}_1,\tilde{x}_2)) \in \delta((x_1,x_2),s)$$

is: given that the node is in state $s$ and it has read $(x_1,x_2)$ on its input lines, the node can "fire" and write $(\tilde{x}_1,\tilde{x}_2)$ on its output lines and, moreover, the automaton comes in a new state $\tilde{s}$.

Data flow nets are data driven. Our model has to capture this. We put a restriction on our automata. It should not be possible for a automaton to neglect some input.

*Restriction:* whenever there is more input available on a tape from which an automaton reads, it will read this information and use it to fire in a *finite* time.

The next step is the definition of the operational semantics. First we introduce the syntax of a data flow net

Definition 2.1. (Syntax of data flow nets)

$$t ::= <d_1, \ldots, d_k>\{i_1{:}j_1, \ldots, i_n{:}j_n\} : k \geq 1, n \geq 0$$

A net consists of a number of basic nodes. The input lines of the nodes are numbered from left to right. The same applies to the output lines. Lines can be connected. This is indicated by the integers between the braces. As an example: {1:2,3:4} means that the first input line is connected to the second output line and the third input line to the fourth output line. Without loss of generality we require that an input line is at most connected to one output line and that an output line is at most connected to one input line. If we have such a data flow net we can distinguish three kinds of lines:

1. input lines of nodes that are not connected to any output lines,
2 input lines of nodes that are connected to output lines,
3. output lines of nodes that are not connected.

In the context of a net, the first kind of lines are called the input lines of a net, the second kind the feed back lines and the third kind are called the output lines of a net.

This definition of the syntax of data flow nets may be to restrictive at first sight. The restriction that only one input line can be connected to one output line is no real restriction. We can use split nodes: nodes with one input which copy all incoming tokens to all output lines. We can use nodes with zero outputs to hide some lines from the outside world. The syntax is not compositional. We use here this syntax because it is easy to associate automata with this kind of networks. In section 3 we will use a different syntax. The nets we use here will be the normal forms of the nets of section 3.

Given a net, we associate a tape with each channel in the network. For every node in the network we have an automaton. These automata will work on the tapes. With each input line of the network a read only tape is associated, and with each output line a write only tape. On the tapes which correspond with feed back lines both reading and writing is possible. It is impossible to read something before it has been written. Stated in a different manner: a read head on a feed back tape is always behind a write head on the same tape. So the execution of a net

$$t \equiv <d_1, \ldots, d_k>\{i_1{:}j_1, \ldots, i_n{:}j_n\}$$

can be viewed as the $k$-automata $d_1, \ldots, d_k$ working simultaneously on tapes.

Example 2.1: The net

$$t \equiv <merge,split,plus\,1>\{2{:}4,3{:}1,4{:}3\}$$

as pictured in figure 2.2,

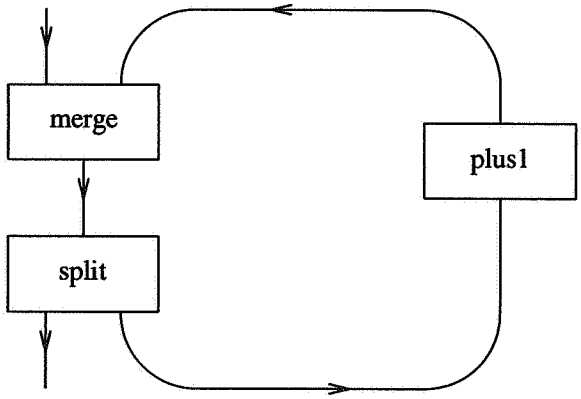


FIGURE 2.2

can be given an operational intuition by considering it as 3 automata, writing and reading on 5 different tapes, as is shown in figure 2.3.
The operational semantics $\mathcal{O}$ describes the behavior of a network as a function. This function takes as input a tuple of words. This tuple represents the input on the tapes which are associated with the input lines of the network. The words can be infinite. The operational semantics maps this tuple to a set of tuples of words. Each tuple represents a possible output on the output tapes after a run of the automata. Such a run can be infinite. We can obtain several alternatives because our nodes can be nondeterministic. Note that the operational semantics does not deliver the contents of the feed back tapes. If a network $t$ has $n$ inputs and $m$ outputs then

$$\mathcal{O}(t) \in Trace^{n:m} =^{def} (A^{\infty})^n \rightarrow \mathcal{P}((A^{\infty})^m).$$

Let $\chi$ be a typical element of $Trace^{n:m}$.

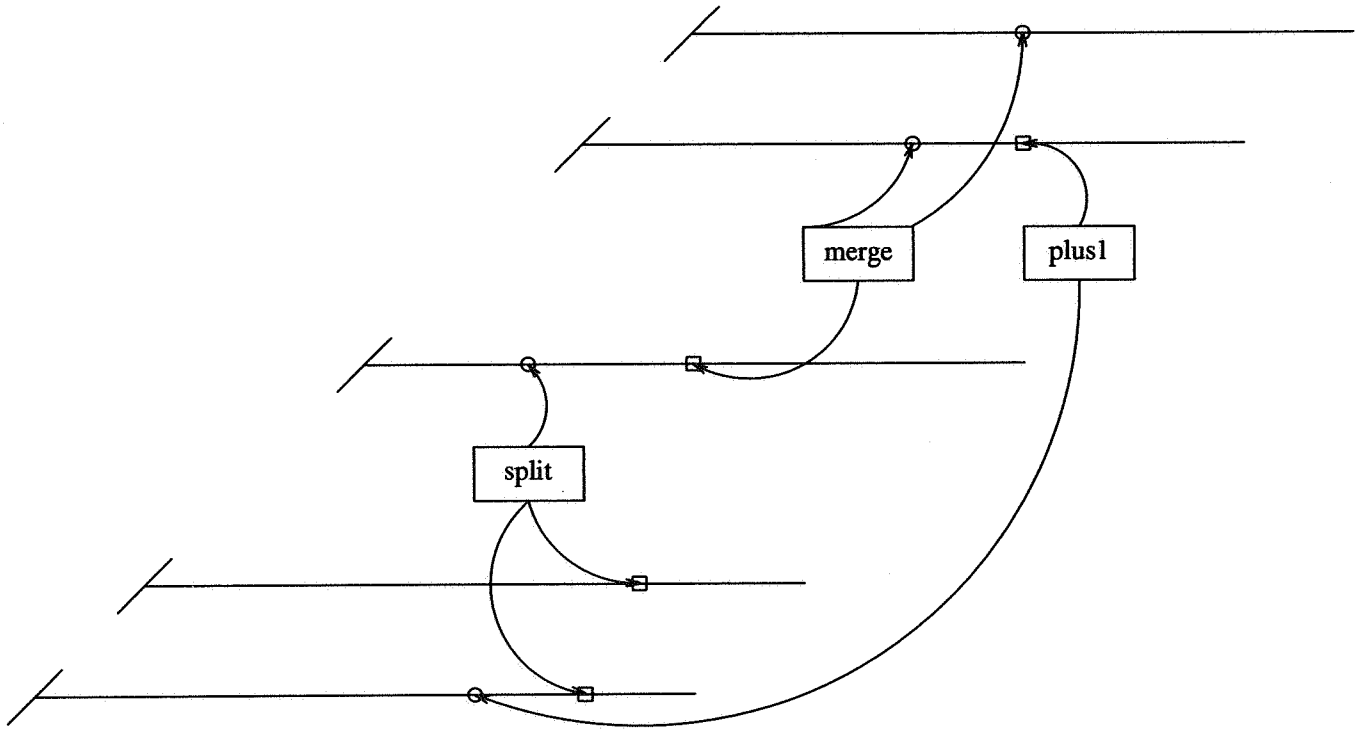


FIGURE 2.3

## 3. DENOTATIONAL SEMANTICS

In this section we define a denotational semantics for data flow nets. This semantics should be compositional. In the previous section we showed that the operational semantics was not compositional. The easiest way to obtain a compositional semantics is to define it in a compositional way. We introduce again data flow nets, but in a different way. They are build up in a compositional way:

Definition 3.1. (Syntax of data flow nets)
*Let t be a typical element of the set Net of data flow nets, which is defined with the following BNF grammar:*

$$\begin{aligned} t ::= \;& d : d \in Node \\ & \mid < t_1, \ldots, t_n > : n \geq 1 \\ & \mid t_1 \{i_1{:}j_1, \ldots, i_n{:}j_n\} : n \geq 1 \end{aligned}$$

Each net $t$ has a normal form $NF(t)$. This normal form is an element of

$$\bigcup_{n,m} \{ <d_1, \ldots, d_n> \{i_1{:}j_1, \ldots, i_m{:}j_m\} : d_1, \ldots, d_n \in Node \}.$$

This normal form relates the "compositional" nets with the nets as defined in section 2. A formal definition of the normal form of a net will be given in section 4. In that section we look how the operational and denotational semantics are related.

With the help of this compositional definition of nets we define the denotational semantics. This definition consists of four steps:
1. introduction of the semantic domains,
2. giving a meaning to basic nodes $d$,
3. defining the semantical equivalent of the tupling operator,

4. and the modeling of feed back loops.

We start by giving the semantic domains.

Definition 3.2. (Semantic Domains)
*Let Dom be the set of infinite sequences of finite words of tokens. An element* $<x_1,x_2,x_3, \ldots >$ *of Dom can be considered as a function which is an element of* $\mathbb{N} \to A^*$. *The integer i is mapped to* $x_i$.
*Let, for each* $n \in \{0,1,2, \ldots\}$, $Dom^n$ *be the set of n tuples of elements of Dom. Let* $\theta,\Gamma,\Psi$ *be typical elements of these sets. An element* $\theta$ *of* $Dom^n$ *can be seen as a function in* $\{1, \ldots ,n\} \to \mathbb{N} \to A^*$.
*Let, for each* $n,m \in \{0,1,2, \ldots\}$, $Dom^{n:m}$ *be the set* $Dom^n \to \mathcal{P}(Dom^m)$, *where* $\mathcal{P}(\ )$ *denotes "subset of", and let* $\phi$ *be a typical element of these sets.*

*We introduce five operations on elements of Dom.*

$$\theta\downarrow\{k_1, \ldots ,k_l\} = \lambda i \in \{1, \ldots ,l\} \,.\, \lambda j \in \mathbb{N} \,.\, \theta(k_i)(j)$$

$$\theta\uparrow\{k_1, \ldots ,k_l\} = \lambda i \in \{1, \ldots ,n-l\} \,.\, \lambda j \in \mathbb{N} \,.$$
$$< \ldots ,\theta(k_1-1),\theta(k_1+1), \ldots ,\theta(k_l-1),\theta(k_l+1), \ldots >$$

$$\epsilon\Box\theta = \lambda i \in \{1, \ldots ,n\} \,.\, \lambda j \in \mathbb{N} \,.\, \begin{cases} \epsilon & \text{if } j=1 \\ \theta(i)(j-1) & \text{if } j>1 \end{cases}$$

$$\theta\leftarrow(\alpha_i)_i = \lambda i \in \{1, \ldots ,n\} \,.\, \lambda j \in \mathbb{N} \,.\, \theta(i)(\sum_{l<j} \alpha_l + 1) \cdot \cdots \cdot \theta(i)(\sum_{l \leq j} \alpha_l)$$

$$\theta_1 \cdot\cdot \theta_2 = \lambda i \in \{1, \ldots ,n_1+n_2\} \,.\, \lambda j \in \mathbb{N} \,.\, \begin{cases} \theta_1(i)(j) & \text{if } 1 \leqslant i \leqslant n_1 \\ \theta_2(i-n_1)(j) & \text{if } n_1 < i \leqslant n_1+n_2 \end{cases}$$

The second step is the the definition of the semantical function associated with a node *d*. We use again an automaton like model to guide the intuition. Recall that in section 2 we introduced the operational semantics by giving for each node an automaton that could read from input lines and write on output lines. We use the same kind of automata, but the tapes have a different structure: they are elements of *Dom*. We now can say that a head of the automaton is in a position i if it is reading or writing the i-th word of the element of *Dom*. We put a restriction on the positions of the read and write heads.

*Restriction:* the position of a write head should always be greater than or equal to the positions of all read heads.

A automaton associated with a node with two inputs and two outputs is shown in figure 3.3. The read heads have circles and the write heads squares. It is not the case that our automata should write exactly the words of an element of *Dom*. For example, if $<1,1, \ldots >$ is the content of a tape, it is possible that our automaton has written the word 111 an infinite number of times. The boundaries are artificial, and are only used for technical reasons. Note also that it is possible for a read or write head to move on without reading or writing. This is indicated by the "reading" or "writing" of the empty word.

We define for each node *d* a function $\phi_d$ by defining $\overline{\theta} \in \phi_d(\theta)$ if and only if there exist a run of automaton *d* on input tapes with contents $\theta$, such that it writes $\overline{\theta}$ on its output tapes.

The third step is to find the semantical equivalent of the tupling operator.

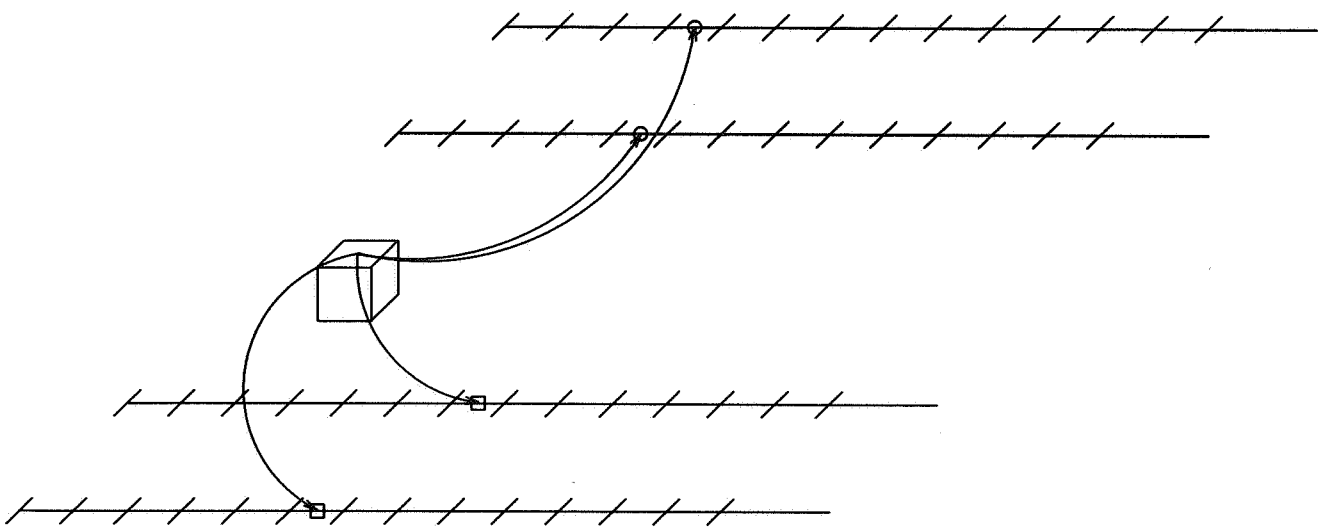

FIGURE 3.3

Definition 3.4. (Semantic tupling)

$$:: : Dom^{n_1:m_1} \times Dom^{n_2:m_2} \to Dom^{n_1+n_2:m_1+m_2}$$

$$\phi_1 :: \phi_2 = \lambda\theta \,.\, \phi_1 \left[ \lambda i \in \{1, \ldots, n_1\} \,.\, \lambda j \in \mathbb{N} \,.\, \theta(i)(j) \right]$$

$$\hat{:}$$

$$\phi_2 \left[ \lambda i \in \{1, \ldots, n_2\} \,.\, \lambda j \in \mathbb{N} \,.\, \theta(n_1+i)(j) \right]$$

where $\hat{:}$ is the extension of concatenation to sets defined by

$$X_1 \,\hat{:}\, X_2 = \bigcup \{ x_1 \cdot x_2 : x_1 \in X_1 \wedge x_2 \in X_2 \}.$$

The fourth step is the modeling of the feed back loop. In the definition we required that on a feed back line the read head was not as far as the write head. We would like to use the structure of elements of *Dom* to formulate such a restriction.
A first guess would be that $\overline{\theta}$ is the result of input $\theta$ iff we have a run of our automaton on input $\theta$ that delivers $\overline{\theta}$ on its output lines, and, moreover, where on all feed back lines we have that the position of a write head is always greater than the position of a read head.
This guess turns out to be to restrictive. It should be the case that there exist finer partitions of $\theta$ and $\overline{\theta}$ such that we have a run of the automaton on these partitions in which the condition above is satisfied.
We have to be a bit more careful when we start the computation. When we start our automaton on a feed back line we first put the write head on the tape, and only after it has left the first position we place the read head on the tape. In the definition of $\mathfrak{D}$ this restriction is formulated with the help of the $\epsilon\Box$ operator. Suppose there exist a run of automata where we have two tapes with contents $\theta_1$ and $\theta_2$ that satisfy $\theta_1 = \epsilon\Box\theta_2$. Assume that the first tape is write only and the second tape is read only. In a run of the automata we have that the the positions of the write heads are always greater or equal to the positions of every read head. If we place the two heads on one tape, we can get as contents $\theta_1$ and by our $\epsilon$ shift it is guaranteed that the position of the read head is always less than the position of the write head.
A typical situation of a network with one node, one input line, one output line and one feed back line

is shown in figure 3.5.

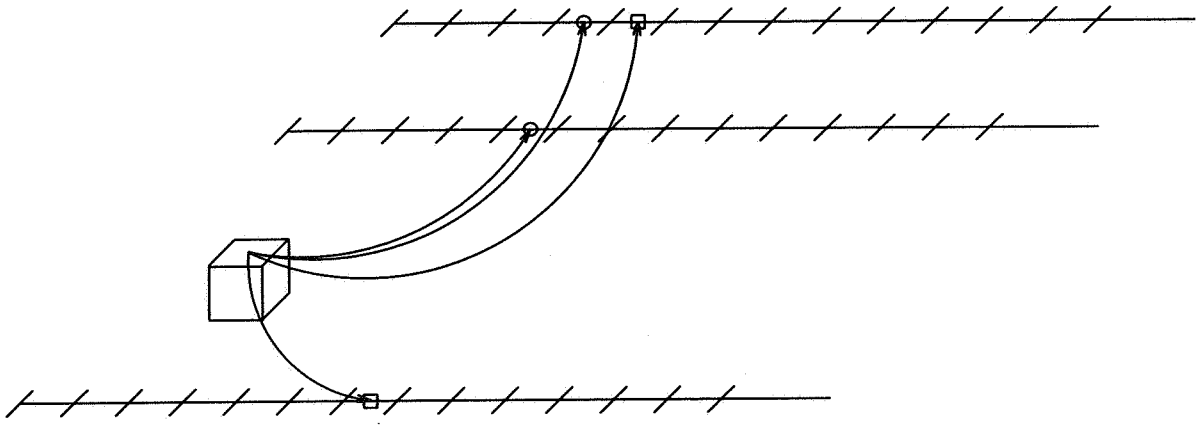

FIGURE 3.5

Now we are prepared to give the definition of the denotational semantics.

Definition 3.6 (Compositional Semantics)
*Let*

$$\mathfrak{D} : Net \rightarrow \bigcup \{Dom^{n:m} : n,m \in \{0,1,2,\ldots\}\}$$

*be recursively defined by*

$$D(d) = \phi_d$$

$$D(< t_1,\ldots,t_n >) = D(t_1) :: \cdots :: D(t_n)$$

$$D(t\ \{i_1{:}j_1,\ldots,i_n{:}j_n\}) =$$

$$\lambda\theta\,.\,\{\,\overline{\theta} : (\exists\theta_1\theta_2)[\theta_1 \in D(t)(\theta_2) \wedge \theta_2{\downarrow}\{i_1,\ldots,i_n\} = \epsilon \,\square\, (\theta_1 \downarrow \{j_1,\ldots,j_n\}) \wedge$$

$$(\exists(\alpha_i)_i)\,[(\theta_1 \uparrow \{j_1,\ldots,j_n\}){\hookleftarrow}(\alpha_i)_i = \overline{\theta} \wedge (\theta_2 \uparrow \{i_1,\ldots,i_n\}){\hookleftarrow}(\alpha_i)_i = \theta\,]\,]$$

4. RELATION BETWEEN OPERATIONAL AND DENOTATIONAL SEMANTICS

In this section we investigate the relation between the semantics of nets which where given in section 2 and 3. The operational semantics was defined on nets of a different form. First we define the normal form of nets which where given in section 3. We show that the denotational semantics of a net and its normal form are the same. In the last part of this section we give an abstraction operator $\alpha$. This operator relates the two semantics: $\mathfrak{O}(t) = \alpha(\mathfrak{D}(t))$ for nets $t$ which are in normal form.

Definition 4.1 (Normal Form)
*Define the normal from a net recursively as follows:*

1. $NF(d) = d$
2. $NF(< t_1,\ldots,t_n >) =$

$$<d_{11},\ldots,d_{1m_1},\ldots,d_{n1},\ldots,d_{nm_n}>$$

$$\{i_{11}{:}j_{11},\ldots,i_{1k_1}{:}j_{1k_1},$$

$$(\alpha_1+i_{21}):(\beta_1+j_{21}),\ldots,(\alpha_1+i_{2k_2}):(\beta_1+j_{2k_2})$$

$$,\ldots,$$

$$(\alpha_1+\cdots+\alpha_{n-1}+i_{n1}):(\beta_1+\cdots+\beta_{n-1}+j_{n1})$$

$$,\ldots,$$

$$(\alpha_1+\cdots+\alpha_{n-1}+i_{nk_n}):(\beta_1+\cdots+\beta_{n-1}+j_{nk_n})\}$$

if, for $l \in \{1,\ldots,n\}$

$$NF(t_l) = <d_{l1},\ldots,d_{lm_l}>\{i_{l1}{:}j_{l1},\ldots,i_{lk_l}{:}j_{lk_l}\} \text{ and } t_l \in Net^{\alpha_l:\beta_l}.$$

3. $NF(t\ \{i_1{:}j_1,\ldots,i_n{:}j_n\}) =$

$$<d_1,\ldots,d_{n_1}>$$

$$\{i_{11}{:}j_{11},\ldots,i_{1k_1}{:}j_{1k_1},(\alpha_1+i_1):(\beta_1+j_1),\ldots,(\alpha_n+i_n):(\beta_n+j_n)\}$$

if

$$NF(t) = <d_1,\ldots,d_{n_1}>\{i_{11}{:}j_{11},\ldots,i_{1k_1}{:}j_{1k_1}\}$$

where, for $l \in \{1,\ldots,n\}$

$$\alpha_l = \#\{i : i \in \{i_{11},\ldots,i_{1k_1}\} \wedge i \leqslant i_l\},$$

$$\beta_l = \#\{j : j \in \{j_{11},\ldots,j_{1k_1}\} \wedge j \leqslant j_l\}.$$

Example: $NF((d\{1{:}1\})\{1{:}1\}) = d\{1{:}1,2{:}2\}$.

We give two properties of our denotational semantics. We use the two properties in the proof of $\mathfrak{N}(t) = \mathfrak{N}(NF(t))$. The first property states that, if a $\overline{\theta}$ is a possible tuple of outputs on input $\theta$, we can add empty words ($\epsilon$) at corresponding places in both $\theta$ and $\overline{\theta}$.

Lemma 4.2.

$$(\forall n,m \in \{1,2,3,\ldots\}\ \forall t \in Net^{n:m}\ \forall \theta,\theta' \in Dom^n\ \forall \overline{\theta},\overline{\theta}' \in Dom^m\ \forall k \in \{0,1,2,\ldots\})[$$

$$\overline{\theta} \in \mathfrak{N}(t)(\theta)$$

$$\wedge$$

$$\theta' = \lambda i\,.\,\lambda j\,.\begin{cases}\theta(i)(j) & \text{if } j<k\\ \epsilon & \text{if } j=k\\ \theta(i)(j-1) & \text{if } j>k\end{cases}$$

$$\wedge$$

$$\overline{\theta}' = \lambda i\,.\,\lambda j\,.\begin{cases}\overline{\theta}(i)(j) & \text{if } j<k\\ \epsilon & \text{if } j=k\\ \overline{\theta}(i)(j-1) & \text{if } j>k\end{cases}$$

$$\Rightarrow$$

$$\bar{\theta}' \in \mathfrak{D}(t)(\theta') \ ]$$

The second property is about what happens if we take words together in both input and output. If we do it in the same place, we obtain that the new output is a possible behavior on the new input.

Lemma 4.3.

$$(\forall n,m \in \{1,2,3,\dots\}\ \forall t \in Net^{n:m}\ \forall \theta \in Dom^n\ \forall \bar{\theta} \in Dom^m\ \forall k \in \{1,2,\dots\})[$$

$$\bar{\theta} \in \mathfrak{D}(t)(\theta)$$

$$\wedge$$

$$\theta' = \lambda i\,.\,\lambda j\,.\begin{cases}\theta(i)(j) & \text{if } j<k\\ \theta(i)(k)\cdot\theta(i)(k+1) & \text{if } j=k\\ \theta(i)(j+1) & \text{if } j>k\end{cases}$$

$$\wedge$$

$$\bar{\theta}' = \lambda i\,.\,\lambda j\,.\begin{cases}\bar{\theta}(i)(j) & \text{if } j<k\\ \bar{\theta}(i)(k)\cdot\bar{\theta}(i)(k+1) & \text{if } j=k\\ \bar{\theta}(i)(j+1) & \text{if } j>k\end{cases}$$

$$\Rightarrow$$

$$\bar{\theta}' \in \mathfrak{D}(t)(\theta') \ ]$$

Proofs: With our automata in mind, it is not difficult to see that these two properties hold. For example, if we consider the second lemma, we have to find a division in finite pieces of the words on the tapes, such that it is guaranteed that on feed back lines the read head is always behind the write head. It suffices to do this in the same way as is done for $\theta$ and $\bar{\theta}$ in $\bar{\theta} \in \mathfrak{D}(t)(\theta)$. □

Generalizations of lemma 4.2 and lemma 4.3 also hold. For example, it is possible to take at an infinite number of places words together or to add epsilons.

Theorem 4.5.

$$(\forall t \in Net\ )[\mathfrak{D}(t) = \mathfrak{D}(NF(t))\ ]$$

Proof: Proof is by induction to the structural complexity of t.
($t \equiv d$): The normal form $NF(d)$ equals $d$, so trivially $\mathfrak{D}(t)=\mathfrak{D}(NF(t))$.
($t \equiv <t_1,\dots,t_n>$): By the definition of the denotational semantics and the induction hypothesis we have

$$\mathfrak{D}(<t_1,\dots,t_n>) = \mathfrak{D}(t_1) :: \cdots :: \mathfrak{D}(t_n) = \mathfrak{D}(NF(t_1)) :: \cdots :: \mathfrak{D}(NF(t_n)).$$

Suppose that $\bar{\theta} \in \mathfrak{D}(<t_1,\dots,t_n>)(\theta)$. This implies, by the definition of the normal form,

$$(\exists \theta_1,\dots,\theta_n,\bar{\theta}_1,\dots,\bar{\theta}_n\ )[\ \theta = \theta_1\ (\cdot\cdot)\ \cdots\ (\cdot\cdot)\ \theta_n \wedge \bar{\theta} = \bar{\theta}_1\ (\cdot\cdot)\ \cdots\ (\cdot\cdot)\ \bar{\theta}_n \wedge$$
$$(\forall i \in \{1,\dots,n\})[\bar{\theta}_i \in \mathfrak{D}(NF(t_i))(\theta_i)]].$$

Choose such $\theta_1,\dots,\theta_n,\bar{\theta}_1,\dots,\bar{\theta}_n$. For each $\theta_i$ and $\bar{\theta}_i$ we can find $\Psi_i$ and $\bar{\Psi}_i$ and a sequence of integers $(z_{ij})_j$ such that

$$\Psi_i \hookleftarrow (z_{ij})_j = \theta_i \wedge \bar{\Psi}_i \hookleftarrow (z_{ij})_j = \bar{\theta}_i\,.$$

and there exist, for each $i \in \{1, \ldots, n\}$, a run of the system $NF(t_i)$ on input $\Psi_i$ that delivers as output $\overline{\Psi}_i$. In such a run, the write head on a feed back line is always at least one word further. Define

$$(z_j)_j = (\max_{i \in \{1,\ldots,n\}} z_{ij})_j.$$

Now we add both in $\Psi_i$ and $\overline{\Psi}_i$ empty words, as in lemma 4.2, and we obtain $\Psi_i'$ and $\overline{\Psi}_i'$, which satisfy

$$\Psi_i' \Leftarrow (z_j)_j = \theta_i \wedge \overline{\Psi}_i' \Leftarrow (z_j)_j = \overline{\theta}_i.$$

While $\overline{\Psi}_i' \in \mathfrak{D}(t_i)(\Psi_i')$, we have, by the definition of $\mathfrak{D}$, $\overline{\theta} \in \mathfrak{D}(t)(\theta)$.

Next we prove the other implication. Suppose that $\overline{\theta} \in \mathfrak{D}(NF(t))(\theta)$. This implies that there exist

$$\theta_1, \ldots, \theta_n, \overline{\theta}_1, \ldots, \overline{\theta}_n$$

such that

$$(\forall i \in \{1, \ldots, n\})[\overline{\theta}_i \in \mathfrak{D}(NF(t_i))(\theta_i)\,] \wedge \theta = \theta_1(\cdot\cdot) \cdots (\cdot\cdot)\theta_n \wedge \overline{\theta} = \overline{\theta}_1(\cdot\cdot) \cdots (\cdot\cdot)\overline{\theta}_n$$

Moreover, there exist a sequence $(z_i)_i$ and $\Psi_1, \ldots, \Psi_n, \overline{\Psi}_1, \ldots, \overline{\Psi}_n$ such that

$$(\forall i \in \{1, \ldots, n\})[\Psi_i \Leftarrow (z_i)_i = \theta_i \wedge \overline{\Psi}_i \Leftarrow (z_i)_i = \overline{\theta}_i]$$

and that we have a run of the system $NF(t_i)$, which reads from tapes with contents $\Psi_i$ and writes $\overline{\Psi}_i$ on output tapes. By induction $\overline{\Psi}_i \in \mathfrak{D}(t_i)(\Psi_i)$ for $i \in \{1, \ldots, n\}$, and by the definition of $\mathfrak{D}$

$$\overline{\Psi}_1(\cdot\cdot) \cdots (\cdot\cdot)\overline{\Psi}_n \in \mathfrak{D}(<t_1, \ldots, t_n>)(\Psi_1(\cdot\cdot) \cdots (\cdot\cdot)\Psi_n)$$

and hence, by lemma 4.3,

$$\overline{\theta} \in \mathfrak{D}(t)(\theta).$$

($t \equiv t_1\{i_1{:}j_1, \ldots, i_n{:}j_n\}$): Suppose $\overline{\theta} \in \mathfrak{D}(t_1\{i_1{:}j_1, \ldots, i_n{:}j_n\})(\theta)$. This implies by the definition of the denotational semantics

$$(\exists \Psi' \Gamma')[\ \Psi' \in \mathfrak{D}(t_1)(\Gamma') \wedge \Gamma' \downarrow \{i_1, \ldots, i_n\} = \epsilon\Box(\Psi' \downarrow \{j_1, \ldots, j_n\}) \wedge$$
$$(\exists (z_i)_i)[\ (\Psi' \uparrow \{j_1, \ldots, j_n\}) \Leftarrow (z_i)_i = \overline{\theta} \wedge (\Gamma' \uparrow \{i_1, \ldots, i_n\}) \Leftarrow (z_i)_i = \theta\ ]]$$

Take such $\Psi'$, $\Gamma'$. By the induction hypothesis we have

$$\Psi' \in \mathfrak{D}(t_1)(\Gamma') \Rightarrow \Psi' \in \mathfrak{D}(NF(t_1))(\Gamma').$$

Suppose that

$$NF(t_1) = <d_1, \ldots, d_k>\{\alpha_1, \beta_1, \ldots, \alpha_m{:}\beta_m\}.$$

For this normal form we can use the intuition of the automata. If

$$\Psi' \in \mathfrak{D}(<d_1, \ldots, d_k>\{\alpha_1, \beta_1, \ldots, \alpha_m{:}\beta_m\})\ (\Gamma')$$

then we know that there exist a system of automata $d_1, \ldots, d_k$. For this system it is possible to make a run on input $\Psi''$, which is a refinement of $\Psi'$, such that this run results in output $\Gamma''$. The tuple $\Gamma''$ is a refinement of $\Gamma'$, and moreover, in the same way as $\Psi''$ is a refinement of $\Psi'$: there exist a sequence of integers $(\overline{z}_i)_i$, all greater that zero, such that $\Psi'' \Leftarrow (\overline{z}_i)_i = \Psi'$ and $\Gamma'' \Leftarrow (\overline{z}_i)_i = \Gamma'$. Recall that

$$\Gamma' \downarrow \{i_1, \ldots, i_n\} = \epsilon\Box(\Psi' \downarrow \{j_1, \ldots, j_n\}). \qquad (*)$$

Assume for simplicity that $n = 1$. We have, by (*), that if we concatenate some words of tape $i_1$, this corresponds to a word that is obtained by putting together some words on line $j_1$:

$$\Gamma''(i_1)(\overline{z}_1 + 1) \cdot \cdots \cdot \Gamma''(i_1)(\overline{z}_2) = \Psi''(j_1)(1) \cdot \cdots \cdot \Psi''(j_1)(\overline{z}_1)$$

$$\Gamma''(i_1)(\bar{z}_2+1)\cdot\ \cdots\ \cdot\Gamma''(i_1)(\bar{z}_3) = \Psi''(j_1)(\bar{z}_1+1)\cdot\ \cdots\ \cdot\Psi''(j_1)(\bar{z}_2)$$
$$\ldots$$

It is possible to replace the tape associated with $i_1$ by

$$\lambda j.\begin{cases}\epsilon & \text{if } j = 1\\ \Psi''(j_1)(j-1) & \text{if } j > 1\end{cases}$$

The reading on tape $i_1$ is slowed down. But now we can conclude that we can connect $i_1$ to $j_1$ and we obtain the desired result.

For the other implication, suppose $\bar{\theta} \in \mathcal{D}(NF(t_1\{i_1{:}j_1, \ldots, i_n{:}j_n\}))(\theta)$. We have a run of the system $NF(t)$, such that $\Psi$ is input on the input tapes and $\bar{\Psi}$ is written on the output tapes. There is also a sequence $(z_i)_i$ such that $\theta = \Psi{\leftarrow}(z_i)_i$ and $\bar{\theta} = \bar{\Psi}{\leftarrow}(z_i)_i$. We break the connections $i_1{:}j_1, \ldots, i_n{:}j_n$, by the addition of $n$ new input and $n$ new output lines. We know that there exist a run of the system $t_1$ on these tapes. In this run the contents of output $j_l$ are the contents of input $i_l$ with one shift to right (one empty word added at the beginning). By induction, applied to $t_1$, we know that $\mathcal{D}(NF(t_1)) = \mathcal{D}(t_1)$. By definition of $\mathcal{D}$

$$\bar{\Psi} \in \mathcal{D}(t)(\Psi)$$

and by lemma 4.3

$$\bar{\theta} \in \mathcal{D}(t)(\theta).\square$$

We continue with the definition of our abstraction operator.

Definition 4.6: *Let* $\theta \in Dom^n$ *and let* $\phi \in Dom^{n:m}$. *We define*

$$\theta[j] = \lambda i \in \{1, \ldots, n\}\ .\ \theta(i)(1)\cdot\ \cdots\ \cdot\theta(i)(j)$$
$$\theta[\infty] = \lim_{j\to\infty}\theta[j]$$

*and*

$$\alpha : Dom^{n:m} \to Trace^{n:m}$$

*by*

$$\alpha(\phi) = \lambda\chi\ .\ \bigcup\{\bar{\chi} : (\exists\theta\bar{\theta})[\ \theta[\infty] = \chi \wedge \bar{\theta}[\infty] = \bar{\chi} \wedge \bar{\theta} \in \phi(\theta)\ ]\ \}$$

Theorem 4.7. $(\forall t \in Net)[\mathcal{O}(NF(t)) = \alpha(\mathcal{D}(NF(t))\ ]$
Proof: consider a run of the automaton on tapes which have contents in *Dom*. An element of *Dom* can be seen as a word which is cut in finite pieces. These pieces do not influence what can be written on or read of tapes. If we abstract from this timing information, it is clear that we get a run of automata which work on tapes with contents in $A^\infty$. □

## 5. Fully Abstractness of the Denotational Semantics

In this section we show that the denotational semantics is fully abstract with respect to the equivalence generated by the operational semantics. First we define the notion of a context. A context is a data flow net with a hole in it. In this hole we can put a net.

Definition 5.1. (Context)

$$C() ::= d : d \in Node$$

$$
\begin{aligned}
&| \ 0 \\
&| \ <t_1, \ldots, t_k, C(), t_{k+1}, \ldots, t_n> : n \geq 1, \ k \geq 0 \\
&| \ C()\{i_1{:}j_1, \ldots, i_n{:}j_n\} : n \geq 1
\end{aligned}
$$

With $Net^{n:m}$ we denote the subset of *Net* of nets with $n$ inputs and $m$ outputs. With $Context^{n:m}$ we denote the subset of *Context* of all contexts for nets with $n$ inputs and $m$ outputs. With the help of the notion of context we can formulate the fully abstractness condition.

Definition 5.2. (Fully Abstract)

$$(\forall n,m \in \{1,2,3, \ldots\})[$$
$$(\forall t_1,t_2 \in Net^{n:m})[\mathcal{D}(t_1) = \mathcal{D}(t_2) \Leftrightarrow (\forall C() \in Context^{n:m})[\mathcal{O}(C(t_1)) = \mathcal{O}(C(t_2))]]]$$

We prove that this condition holds in two steps. Theorem 5.3 and 5.4 together imply that the fully abstract condition holds.

Theorem 5.3.

$$(\forall n,m \in \{1,2,3, \ldots\})[$$
$$(\forall t_1,t_2 \in Net^{n:m})[\mathcal{D}(t_1) = \mathcal{D}(t_2) \Rightarrow (\forall C() \in Context^{n:m})[\mathcal{O}(C(t_1)) = \mathcal{O}(C(t_2))]]]$$

Proof: if $\mathcal{D}(t_1) = \mathcal{D}(t_2)$, then also, by the compositionality of $\mathcal{D}$, $\mathcal{D}(C(t_1)) = \mathcal{D}(C(t_2))$. We have that $\mathcal{O} = \alpha \circ \mathcal{D}$, and hence $\mathcal{O}(C(t_1)) = \mathcal{O}(C(t_2))$. □

Theorem 5.4.

$$(\forall n,m \in \{1,2,3, \ldots\})[$$
$$(\forall t_1,t_2 \in Net^{n:m})[\mathcal{D}(t_1) \neq \mathcal{D}(t_2) \Rightarrow (\exists C() \in Context^{n:m})[\mathcal{O}(C(t_1)) \neq \mathcal{O}(C(t_2))]]]$$

Proof: If $\mathcal{D}(t_1) \neq \mathcal{D}(t_2)$ then we may assume that we can find $\theta$ and $\overline{\theta}$ such that

$$\overline{\theta} \in \mathcal{D}(t_1)(\theta) \wedge \overline{\theta} \notin \mathcal{D}(t_2)(\theta).$$

If we can not find such $\theta$ and $\overline{\theta}$ we reverse the role of $t_1$ and $t_2$.
In the rest of the proof we take $n = m = 1$. It is not difficult to extend the proof for nets with more inputs and/or outputs.
If $n = 1$ then $\theta \in \{1\} \rightarrow \{1,2,3, \ldots\} \rightarrow A^*$ and if $m = 1$ then $\overline{\theta} \in \{1\} \rightarrow \{1,2,3, \ldots\} \rightarrow A^*$. Define the context C() to be $<t, ()>\{2{:}3, 3{:}2\}$, where $t$ is the net pictured in figure 5.5. In the net $t$ we find the following nodes:
*merge:* a node that merges its two inputs,
*split:* a node that, when it receives a token, copies it and outputs the two identical tokens, one on each output line,
*tt:* a node that tags tokens with a color, such that the difference between a tagged token and a non tagged token can be observed,
*rtt:* when it receives a tagged token, it outputs nothing, and when it receives a non tagged token, it will output it.
Define

$$\tilde{\theta} : \{1\} \rightarrow \{1,2,3, \ldots\} \rightarrow A^*$$

by

$$\tilde{\theta}(1)(i) = \theta(1)(i) \cdot tag(\overline{\theta}(1)(i)).$$

and
$$\begin{cases} y = \theta(1)(1)\cdot\theta(1)(2)\cdot\theta(1)(3)\cdot\ \cdots \\ \overline{y} = \overline{\theta}(1)(1)\cdot\overline{\theta}(1)(2)\cdot\overline{\theta}(1)(3)\cdot\ \cdots \\ \tilde{y} = \tilde{\theta}(1)(1)\cdot\tilde{\theta}(1)(2)\cdot\tilde{\theta}(1)(3)\cdot\ \cdots \end{cases}$$

We show
(i) $\tilde{y} \in \mathcal{O}(C(t_1))(y)$
(ii) $\tilde{y} \notin \mathcal{O}(C(t_2))(y)$

(i) We know that $\overline{\theta} \in \mathcal{D}(t_1)(\theta)$. From the definition of $\mathcal{D}$ we conclude that there exist $\Psi$ and $\overline{\Psi}$ such that there exist a run of the system $NF(t_1)$ on input $\Psi$ which delivers as output $\overline{\Psi}$. There exists also a sequence $(z_i)_i$ such that $\theta\leftarrow(z_i)_i = \Psi$ and $\overline{\theta}\leftarrow(z_i)_i = \overline{\Psi}$. We number the channels of $C(t_1)$ as is indicated in figure 5.5. We construct runs of automata associated with the four nodes of our context.

*merge* node:

left input line

$<\theta(1)(1),\epsilon, \ldots, \epsilon,$ ( $3+z_1$ *epsilons* )
$\theta(1)(2),\epsilon, \ldots, \epsilon,$( $3+z_2$ *epsilons* )
$\theta(1)(3),\epsilon, \ldots, \epsilon,$
$\ldots >$

right input line

$<\epsilon,\epsilon,\epsilon,tag(\overline{\Psi}(1)(1)), \ldots, tag(\overline{\Psi}(1)(z_1)),\epsilon,\epsilon,\epsilon,\epsilon,$
$tag(\overline{\Psi}(1)(z_1+1)), \ldots, tag(\overline{\Psi}(1)(z_2)),\epsilon,\epsilon,\epsilon,\epsilon,$
$\ldots >$

output line

$<\theta(1)(1),\epsilon,\epsilon,\epsilon,tag(\overline{\Psi}(1)(1)),, \ldots, tag(\overline{\Psi}(1)(z_1)),$
$\theta(1)(2),\epsilon,\epsilon,\epsilon,tag(\overline{\Psi}(1)(z_1+1)), \ldots, tag(\overline{\Psi}(1)(z_2)),$
$\ldots >$

*dup* automaton

input line

$<\epsilon,\theta(1)(1),\epsilon,\epsilon,\epsilon,tag(\overline{\Psi}(1)(1)),, \ldots, tag(\overline{\Psi}(1)(z_1)),$
$\theta(1)(2),\epsilon,\epsilon,\epsilon,tag(\overline{\Psi}(1)(z_1+1)), \ldots, tag(\overline{\Psi}(1)(z_2)),$
$\ldots >$

left output line

$<\epsilon,\theta(1)(1),\epsilon,\epsilon,\epsilon,tag(\overline{\Psi}(1)(1)), \ldots, tag(\overline{\Psi}(1)(z_1)),$
$\theta(1)(2),\epsilon,\epsilon,\epsilon,tag(\overline{\Psi}(1)(z_1+1)), \ldots, tag(\overline{\Psi}(1)(z_2)),$
$\ldots >$

right output line

$$<\epsilon,\theta(1)(1),\epsilon,\epsilon,\epsilon,tag(\overline{\Psi}(1)(1)),\ldots,tag(\overline{\Psi}(1)(z_1)),$$
$$\theta(1)(2),\epsilon,\epsilon,\epsilon,tag(\overline{\Psi}(1)(z_1+1)),\ldots,tag(\overline{\Psi}(1)(z_2)),$$
$$\ldots>$$

*rtt* automaton

input line

$$<\epsilon,\epsilon,\theta(1)(1),\epsilon,\epsilon,\epsilon,tag(\overline{\Psi}(1)(1)),\ldots,tag(\overline{\Psi}(1)(z_1)),$$
$$\theta(1)(2),\epsilon,\epsilon,\epsilon,tag(\overline{\Psi}(1)(z_1+1)),\ldots,tag(\overline{\Psi}(1)(z_2)),$$
$$\ldots>$$

output line

$$<\epsilon,\epsilon,\Psi(1)(1),\ldots,\Psi(1)(z_1),\epsilon,\epsilon,\epsilon,\epsilon,$$
$$\Psi(1)(z_1+1),\ldots,\overline{\Psi}(1)(z_2),\epsilon,\epsilon,\epsilon,\epsilon,$$
$$\ldots>$$

*tt* automaton

input line

$$<\epsilon,\epsilon,\epsilon,\overline{\Psi}(1)(1),\ldots,\overline{\Psi}(1)(z_1),\epsilon,\epsilon,\epsilon,\epsilon,$$
$$\overline{\Psi}(1)(z_1+1),\ldots,\overline{\Psi}(1)(z_2),\epsilon,\epsilon,\epsilon,\epsilon,$$
$$\ldots>$$

output line

$$<\epsilon,\epsilon,\epsilon,tag(\overline{\Psi}(1)(1)),\ldots,tag(\overline{\Psi}(1)(z_1)),\epsilon,\epsilon,\epsilon,\epsilon,$$
$$tag(\overline{\Psi}(1)(z_1+1)),\ldots,tag(\overline{\Psi}(1)(z_2)),\epsilon,\epsilon,\epsilon,\epsilon,$$
$$\ldots>$$

From $\Theta = \alpha\circ\mathfrak{D}$ we derive $\tilde{y} \in \Theta(C(t_1))(y)$.

(ii) Suppose $\tilde{y} \in \Theta(C(t_2))(y)$. While $\Theta = \alpha\circ\mathfrak{D}$, there exist $\Psi_1$ and $\Psi_2$ such that $\Psi_1[\infty] = y$, $\Psi_2[\infty] = \tilde{y}$, $\Psi_2 \in \mathfrak{D}(C(t_2))(\Psi_1)$ and there exist a run of the system $NF(C(t_2))$ on input $\Psi_1$ such that $\Psi_2$ is delivered as output.
We take the numbering of the lines as in figure 5.5. We associate with lines $\{1,2,3,4,5\}$ a contents $\Gamma_i$ and $\overline{\Gamma}_i \in \{1\}\rightarrow\{1,2,3,\ldots\}\rightarrow A^*$. A run of system $NF(C(t_2))$ consists of runs of the automata associated with nodes. For each of the lines $1,\ldots,5$ there is an automaton that writes on it and one that reads from it. If for a certain line $\overline{\Gamma}_i$ is written, and $\Gamma_i$ is read, then we have by the definition of the denotational semantics that $\epsilon\Box\overline{\Gamma}_i = \Gamma_i$. From this we derive that for all i $\Gamma_i[\infty] = \overline{\Gamma}_i[\infty]$. Because $\Psi_1[\infty] = y$ and $\Psi_2[\infty] = \tilde{y}$ we know that

$$\Gamma_1[\infty] = \overline{y},$$
$$\Gamma_2[\infty] = tag(\overline{y}),$$
$$\Gamma_3[\infty] = \tilde{y},$$
$$\Gamma_4[\infty] = \tilde{y},$$
$$\Gamma_5[\infty] = y.$$

We define ten infinite sequences of integers: $(k_i)_i$, $(l_i)_i$, $(m_i)_i$, $(n_i)_i$, $(o_i)_i$. $(\overline{k}_i)_i$, $(\overline{l}_i)_i$, $(\overline{m}_i)_i$, $(\overline{n}_i)_i$, $(\overline{o}_i)_i$.

$$\Gamma_1[k_i-1] < \overline{\theta}[i] \leqslant \Gamma_1[k_i]$$
$$\overline{\Gamma}_1[\overline{k}_i-1] < \overline{\theta}[i] \leqslant \overline{\Gamma}_1[\overline{k}_i]$$
$$\Gamma_2[l_i-1] < tag(\overline{\theta}[i]) \leqslant \Gamma_2[l_i]$$
$$\overline{\Gamma}_2[\overline{l}_i-1] < tag(\overline{\theta}[i]) \leqslant \overline{\Gamma}_2[\overline{l}_i]$$
$$\Gamma_3[m_i-1] < \tilde{\theta}[i] \leqslant \Gamma_3[m_i]$$
$$\overline{\Gamma}_3[\overline{m}_i-1] < \tilde{\theta}[i] \leqslant \overline{\Gamma}_3[\overline{m}_i]$$
$$\Gamma_4[n_i-1] < \tilde{\theta}[i] \leqslant \Gamma_4[n_i]$$
$$\overline{\Gamma}_4[\overline{n}_i-1] < \tilde{\theta}[i] \leqslant \overline{\Gamma}_4[\overline{n}_i]$$
$$\Gamma_5[o_i] \leqslant \theta[i] < \Gamma_5[o_i+1]$$
$$\overline{\Gamma}_5[\overline{o}_i] \leqslant \theta[i] < \overline{\Gamma}_5[\overline{o}_i+1]$$

Remark that the conditions on $(o_i)_i$ and $(\overline{o}_i)_i$ are somewhat different from the other eight. By the $\epsilon$ shift we have for all i

$$k_i = \overline{k}_i+1$$
$$l_i = \overline{l}_i+1$$
$$m_i = \overline{m}_i+1$$
$$n_i = \overline{n}_i+1$$
$$o_i = \overline{o}_i+1$$

and from properties of the nodes we derive

$$k_i \leqslant \overline{l}_i$$
$$l_i \leqslant \overline{m}_i$$
$$m_i \leqslant \overline{n}_i$$
$$n_i-1 \leqslant \overline{o}_i$$

so for all i we have $\overline{k}_i < o_i$. We have that

$$\Gamma_5 \in \mathfrak{D}(t_2)(\overline{\Gamma}_1)$$

and from this and $\overline{k}_i < o_i$ we derive

$$\overline{\theta} \in \mathfrak{D}(t_2)(\theta).$$

Contradiction □.

## 6. Conclusion

The denotational framework presented in this paper is fully abstract with respect to the operational semantics. Our semantics can handle all nodes that can be described by the specification functions. This set of nodes includes some kinds of fair merge nodes. In section 2 we required that we can not forget any input and we can use (as internal states of nodes) oracles. One of the advantages of our model is, in our opinion, that we have on the one hand an intuitive model based on automata and on the other hand a formal calculus. On the calculus side we have domains, operations on them (like the

$\epsilon$ shift) and it is possible to model the feed back loop by a fixed point of a higher order function. This approach was taken in [Kok 1986]. The framework presented there is compositional, but not fully abstract. It is possible to present the system of this paper in a similar style, without the intuition of automata. On the other hand, the automata theoretic approach simplifies proofs and provides insights into the model. The difficulty is that it applies directly only to nets in normal form.

The problems with finding a compositional semantics for nondeterministic data flow were first discussed in [Brock & Ackerman 1981]. The non-compositionality of a semantics based on words of tokens is often called the Brock-Ackerman anomaly. In the same paper they propose a compositional semantics. This semantics is based on scenarios. A scenario is a graph like structure. Such structures contain enough information for the compositionality. This framework is not fully abstract.
A different approach, for example taken by [Broy 1984,1985] and [Park 1983], is the use of oracles. An oracle tells a nondeterministic node which choices it has to make. Given an oracle, a node behaves deterministically. It is possible to apply Kahn's method to a net, given oracles for all nondeterministic nodes. To obtain the semantics of a net take the union over all possible oracles. The frameworks based on oracles often use more sophisticated orderings on histories than just the prefix order. In this way it is possible to obtain a compositional semantics.
Algebraic treatments of nondeterministic nets are given in [Back & Mannila 1982] and in [Bergstra & Klop 1983].
In [Staples & Nguyen 1985] it is claimed that all of the models they know of are not fully abstract.

Acknowledgements: first of all, we would like to thank Jaco de Bakker, for the support and the guidance during the writing of this paper. A paper of him ([de Bakker et al 1985]) served as a starting point for our investigation. Some of the notation is borrowed from that paper, and an equivalent of theorem 4.5 (in an order theoretic context) can be found there. The careful reading of the various drafts of this paper and the suggestions to approve them are gratefully acknowledged. Next we would like to thank professor Rozenberg, who suggested the automata-theoretic interpretation of our model, and the other members of LPC kerngroep: professor de Roever, professor de Bakker, Ron Koymans and Iko Keesmaat for their patience and useful suggestions. We are also grateful to the concurrency group on the CWI (Jaco de Bakker, Frank de Boer, Jan Rutten, John-Jules Meyer and Erik de Vink) for their discussions on the contents of this paper.

## References

1. [Abramsky 1983], S. Abramsky, *On Semantic Foundations for Applicative Multiprogramming,* in Proc. 10th ICALP ( J. Diaz ed.), LNCS 154, Springer, 1983, pp. 1-14.
2. [Abramsky 1984], S. Abramsky, *Reasoning About Concurrent Systems,* in Distributed Computing ( F.B. Chambers, D.A. Duce, G.P. Jones eds.), 1984.
3. [Arnold 1981], A. Arnold, *Semantique des Processus Communicants,* RAIRO 15 (2), 1981, pp. 103-109.
4. [Back & Mannila 1982], R.J. Back, N. Mannila, *A Refinement of Kahn's Semantics to Handle Nondeterminism and Communication,* in Proc. ACM Symp. on Distributed Comp., Ottawa, 1982, pp.111-120.
5. [de Bakker et al 1985], J.W. de Bakker, J.-J.Ch. Meyer, J. Zucker, *Bringing Color into the Semantics of Nondeterministic Data Flow,* Preprint, Centre for Mathematics and Computer Science, 1985.
6. [Bergstra & Klop 1983], J. Bergstra, J.W. Klop, *Process Algebra for the Operational Semantics of Static Data Flow Networks,* Techn. Report Mathematical Centre IW 222/83, Amsterdam, 1983.
7. [Boussinot 1982], Boussinot, *Proposition de semantique denotationelle pour des reseaux de processus avec operateur de melange equitable,* TCS 18, 1982, pp. 173-206.
8. [Brock & Ackerman 1981], J.D. Brock, W.B. Ackerman, *Scenarios : A Model of Non-*

*determinate Computation,* in Proc. Formalization of Programming Concepts ( J. DIAZ, J. RAMOS eds.), LNCS 107, Springer, 1981, pp.252-259.

9. [BROY 1983], M. Broy, *Fixed Point Theory for Communication and Concurrency,* in Formal Description of Programming Concepts-II ( D. BJØRNER ed.), North-Holland, Amsterdam, 1983, pp. 125-148.
10. [BROY 1984], M. Broy, *Nondeterministic Data Flow Programs : How to avoid the Merge Anomaly,* preprint, Fakultät für Mathematik und Informatik, Universität Passau, 1984.
11. [BROY 1985], M. Broy, *Extensional Behavior of Concurrent, Nondeterministic, Communicating Systems,* in Control Flow and Data Flow : Concepts of Distributed Programming ( M. BROY ed.), pp. 229-277.
12. [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.
13. [KAHN 1974], G. Kahn, *The Semantics of a Simple Language for Parallel Programming,* in Proc. IFIP74, North-Holland, Amsterdam, 1977, pp. 993-998.
14. [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.
15. [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.
16. [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.), LNCS 197, Springer, 1985, pp. 479-496.
17. [KOK 1986], J.N. Kok, *Denotational Semantics of Nets with Nondeterminism,* in Proceedings ESOP 1986, LNCS 213, Springer, pp. 237-250.
18. [KOSINSKI 1978], P.R. Kosinski, *A Straightforward Denotational Semantics for Nondeterminate Data Flow Programs,* in Proc. 5th ACM POPL, 1978, pp. 214-221.
19. [NADLER 1970], S.B. Nadler, *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.
20. [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.
21. [STAPLES & NGUYEN 1985], J.Staples, V.L. Nguyen, *A Fixpoint Semantics for Nondeterministic Dataflow,* Journal of the ACM, 32 (2), 1985, pp. 411-445.