INFORMATION AND COMPUTATION **138**, 49–87 (1997)
ARTICLE NO. IC972658

# On Schematological Equivalence of Partially Interpreted Dataflow Networks

Alexander Rabinovich

*Department of Computer Science, Raymond and Beverly Sackler Faculty of Exact Sciences, Tel Aviv University, Tel Aviv, Israel*

We provide a schematologically complete set of transformations for partially interpreted dataflow nets. This paper extends our results in (*Inform. and Comput.* **124** (1996), 154–167), where a schematologically complete set of transformations was provided for the nets which consist only of uninterpreted nodes and the results for partially interpreted nets were announced without proofs. © 1997 Academic Press

*Contents*

## 1. INTRODUCTION

### 1.1. Summary of Our Previous Results

Consider the net in Fig. 1. It consists of nodes and of directed channels. Channels are partitioned into *internal* (those which connect nodes), *input* (those which enter the net), and *output* (those which exit the net).

Actually, such a net is a piece of syntax. An interpretation assigns meanings to the nodes. The semantics defines what object is assigned to an interpreted net. The nodes of a net are labeled by names. For nodes labeled by the same name an interpretation should assign the same object.

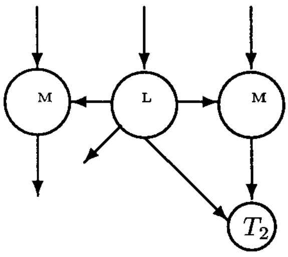


**FIG. 1.** Net $N_1$.

In dataflow, nodes are interpreted by labeled transition systems or in more classical terminology by automata (maybe with an infinite number of states). These automata are working asynchronously and communicate between themselves and the environment by passing data over unbounded FIFO channels. There is an appealing operational semantics which specifies the automaton assigned to an interpreted net as a whole [2, 5, 6, 10, 11, 19].

Consider nets $N_1$ and $N_2$ in Fig. 1 and in Fig. 2. These nets are schematologically equivalent, i.e., they have the same I(nput)–O(utput) behavior under all interpretations. Note that $N_2$ has a subnet consisting of the nodes $A$ and $B$. This subnet has neither input nor output channels which are connected to the environment or to other nodes in the net. Following [15], such a subnet is called an isolated subnet. $N_2$ also contains two nodes $T$ which do not have an outgoing channel. Such nodes are called *terminating* nodes.

$N_1$ is obtained from $N_2$ by the following behavior preserving transformations:

$R_1$: Remove an isolated subnet.

$R_2$: Replace a subnet without output channels by a terminating node.

The main results of our previous paper [18] are:

THEOREM (Completeness). *Two nets have the same Input–Output behavior under all dataflow interpretations iff they can be reduced to isomorphic nets by the reduction rules $R_1$ and $R_2$.*

THEOREM (Decidability of Schematological Equivalence). *Input–Output equivalence of nets under all dataflow interpretations is decidable in polynomial time.*

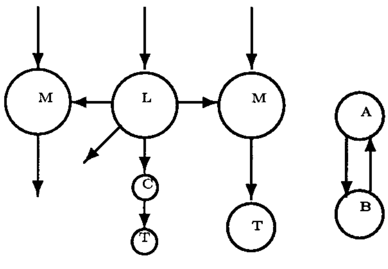


**FIG. 2.** Net $N_2$.

## *1.2. Contribution of This Paper*

This paper extends the above theorems to partially interpreted dataflow nets. Such nets in addition to uninterpreted nodes can contain primitive nodes which are interpreted always in the same way. The following primitive nodes (see Fig. 3) play an important role in dataflow:

*Primitive Nodes.*

- Terminating node. It has no output channels and a number of input channels. It consumes the data arriving on its channels. The terminating node with $k$ input channels is denoted by $T_k$.
- Multiplicator. An $n$-plicator has one input channel and $n$ output channels and copies the data received over the input channel to all its output channels. $n$-plicators will be pictured as triangles.
- *Copy*($s$). This node is parameterized by a stream $s$ and has one input and one output channel. It first sends the stream $s$ over its output and then copies the data received over the input to the output.
- *Gen*($s$). This node is parameterized by a stream $s$ and has only one output channel. It generates the stream $s$ on its output channel and then stops.

We will add reductions involving the above mentioned primitive nodes and show that two partially interpreted nets are I–O equivalent under all interpretations iff they can be transformed to isomorphic nets by the primitive reductions and by reductions $R_1$, $R_2$.

## *1.3. Comparison of This Paper and* [18]

The main contribution of this paper is the extension of the results of [18] to the partially interpreted dataflow networks. We emphasized the similarity between these two papers by giving the same formulation to the main theorems. However, these theorems refer here to the *partially interpreted* dataflow nets whereas in [18] they refer to the *uninterpreted* dataflow nets. Even some technical lemmas have almost the same formulation in both papers. This is achieved due to an appropriate conservative extension of notions from [18] to the partially interpreted dataflow nets (see for example the definition of a chain, section 5.1). The results of this paper were announced without proof in [18] .

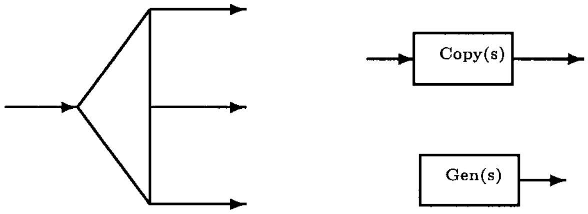


**FIG. 3.** Primitive nodes.

Let us comment on the structure of the proofs for the main theorems stated above.

As usual, the proof of the completeness theorem proceeds in two steps.

*Step* 1. Show the soundness of transformations.

*Step* 2. Find an appropriate characterization for normal form; show that every net is equivalent to a net in normal form; find an interpretation which distinguishes between distinct normal forms.

The following remarks of Girard [7] about current proof theoretical techniques also hold for techniques for proofs of completeness theorems. "The main technical limitation of such methods is... the lack of modularity: in general neighboring problems can be attacked by neighboring methods; but it is only exceptional that one of the problems will be corollary of the other.... Most of the time, a completely new proof will be necessary (but without any new idea). This renders work in the domain quite long and tedious."

For the case of partially interpreted nets step 1 in the proof is based on some ideas which were not required for the uninterpreted case.

Step 2 in both papers is based on the same ideas, however, this step is technically much more complicated for the case of partially interpreted nets.

Our proof of polynomial decidability of schematological equivalence also proceeds in two steps.

*Step* 1. Show that for every net an equivalent net in normal form can be found in polynomial time (Section 3.2).

*Step* 2. Show that it can be checked in polynomial time whether two nets in normal forms are equivalent.

The proof of Step 1 is very different for the case of partially interpreted nets and the uninterpreted nets. However, Step 2 of the proof here is reduced to the corresponding step for the uninterpreted nets and to Lemma 18 which shows that the equivalence between *restricted regular expressions* (see Section 2.1 for their definition) is decidable in polynomial time.

## *1.4. The Organization of the Paper*

The paper is organized as follows. Section 2 is based on the corresponding Section of [18] and describes the syntax of partially interpreted dataflow nets. For an informal presentation of the semantics of dataflow nets we refer the reader to Section 3.2 of [18]. Reduction rules and normalization theorem are given in Section 3. Section 4 gives the soundness theorem and Section 5 gives the completeness theorem. In Section 6 the complexity of schematological equivalence is analyzed. Our work has been very much influenced by Parrow's paper [15]. We refer the reader to [18] for a detailed comparison to Parrow's work [15], a discussion of other related work [4, 20], and a list of open problems.

## 2. SYNTAX OF NETS

We recall below some notions related to the syntax of nets emphasizing the requirement for primitive nodes.

Figure 4 presents the unabbreviated syntax of nets. The net has five nodes. Two of them are labeled by $M$, one by $L$, one by $K$, and one is a terminating node labeled by $T_2$.

A node in the net has several ports. Ports of nodes may be connected by directed links which are called channels. There is always only one channel connected to a port.

A channel is *internal* if it connects two ports; other channels are called *external.* The input (output) channels of a net are the external channels which enter (exit) the ports of the net.

The external channels are labeled by numbers; ports are also labeled by numbers. If a channel enters (exits) a node at port $m$, then the channel is an input (output) channel for the node and $m$ is an input (output) port of the node.

The requirements on labeling of channels, of non-primitive nodes, and of ports are given in section 3.1 of [18]. Below we recall these requirements and state additional requirements on labeling of primitive nodes and their adjacent ports. But first, let us define the restricted regular expressions which will be used as the notations for the parameters of copy and generator nodes.

### *2.1. Restricted Regular Expressions*

Restricted regular expressions over an alphabet $A$ are defined by the grammar $E ::= a \in A \mid \varepsilon \mid EE \mid E^{\omega}$, where $\varepsilon$ is the empty stream, $EE$ is concatenation, and $E^{\omega}$ is infinite iteration.

A restricted regular expression denotes either a finite length word $a_1 a_2 \cdots a_n$ or a quasiperiodic $\omega$-word $a_1 \cdots a_n(b_0, ..., b_m)^{\omega}$. It is clear that an $\omega$-free expression denotes a finite length word. Both words and $\omega$-words over $A$ will be called streams over $A$. Expressions are *equal* if they denote the same stream.

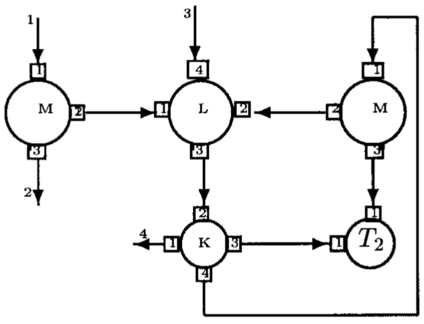


**FIG. 4.** Unabbreviated syntax.

## *2.2. Labeling*

*Labeling of Primitive Nodes.* (1) Terminating node with $k$ input channels is labeled by $T_k$. (2) Multi-plicator node with $k$ output channels is labeled by $k$ and it is called $k$-plicator. (3) The remaining kinds of primitive nodes are copy(s) and generator(s). These nodes are parameterized by a stream. Copy nodes will have labels of the form $Copy(E)$, where $E$ is an $\omega$-free restricted regular expression and generator nodes will have the labels $Gen(E)$, where $E$ is a restricted regular expression. Sometimes we will abbreviate our figures and drop the labels of multi-plicator or terminating nodes.

*Remark.* Our decision to allow the parameters of generator nodes to be infinite streams is not for the sake of generality. The reduction rule $S_6$ given in Section 3.2 reduces a net consisting of copy nodes with finite parameters to a net consisting of generator nodes with infinite quasiperiodic parameters.

*Labeling of Ports, Channels, and Non-primitive Nodes.*

*External Channel Labeling.* Distinct external channels of a net are labeled by distinct numbers.

*Port Labeling of*

*non-primitive nodes.* Distinct ports of a non-primitive node are labeled by distinct numbers.

*primitive nodes.* All output ports of copy, generator and $n$-plicator nodes are labeled by 1. The input ports of copy and $n$-plicator nodes are labeled by 2. All ports of a terminating node are labeled by 1.

*Consistency of Port and Node Labelings.* If two nodes are labeled by the same name, then the sets of numbers assigned to the input ports of these nodes should coincide; also, the sets of numbers assigned to the output ports of the nodes should coincide.

*Remark.* Roughly speaking, the injective labeling of the external channels of a net allows us to distinguish between nets with I–O behaviors $\lambda x.\lambda y.f(x, y)$ and $\lambda x.\lambda y.f(x, y)$. The role of labeling of the ports of non-primitive nodes is similar. A multi-plicator node produces the same output on all its ports. If we had required an injective labeling for output ports of multi-plicator nodes we would have obtained from the net consisting of one 2-plicator two non isomorphic nets with the same I–O behavior. Therefore, our completeness theorem would have failed. The reason to use the non-injective labeling for the input ports of terminating nodes is the same.

## *2.3. Net Isomorphism*

Let us also recall the definition of a net isomorphism.

DEFINITION 1 [18]. Two nets are isomorphic if there exists a bijective map $\phi$ between their nodes such that

1. $\phi$ preserves the labeling of the nodes, i.e., (a) $v$ has a non-primitive label $L$ iff $\phi(v)$ has the label $L$; (b) $v$ is a terminated node with $k$ input ports iff $\phi(v)$ is a terminated node with $k$ input ports; (c) $v$ is a copy (respectively generator) node labeled by $Copy(E_1)$ (respectively by $Gen(E_1)$) iff $\phi(v)$ is labeled by $Copy(E_2)$ (respectively by $Gen(E_2)$) and $E_1$ and $E_2$ denote the same stream.

2. $\phi$ preserves the adjacency relation i.e., there is a channel from a port number $m_1$ of a node $v_1$ to a port $m_2$ of a node $v_2$ if and only if there is a channel from a port $m_1$ of the node $\phi(v_1)$ to a port $m_2$ of the node $\phi(v_2)$

3. $\phi$ preserves the labeling of external channels, i.e., there is an input (output) channel labeled by $a$ which enters (exits) node $v$ at a port number $i$ if and only if there is an input (output) channel labeled by $a$ which enters (exits) the node $\phi(v)$ at a port number $i$.

### 2.4. *Size of a Net*

We define the size of a net $N$ as the number of its nodes plus the number of its channels plus the sum of the lengths of the expressions which appear as parameters of the copy and generator nodes of $N$. In the sequel we consider a number of decision problems for nets. We will measure the complexity of algorithms on nets as the function of the size of input nets.

## 3. REDUCTIONS

In this section we first recall the behavior preserving reduction rule given in [18] and then prove a normal form theorem.

### 3.1. *Subnets*

A subnet of a net $N$ is a subset of the nodes of $N$ together with their ports and the channels which are entering or exiting these nodes. A channel which connects two nodes of a subnet is called an internal channel of the subnet. Let $N'$ be a subnet of $N$. A channel *ch* of $N'$ is an input channel of $N'$ if it enters a node in $N'$ and it is not an internal channel of $N'$. A channel *ch* of $N'$ is an output channel of $N'$ if it exits a node in $N'$ and it is not an internal channel of $N'$. A subnet is *isolated* if it has only internal channels. A subnet is *terminating* if it does not have output channels.

### 3.2. *Reduction Rules*

Every rule $R$ has a form $N_1 \Rightarrow N_2$, where $N_1, N_2$ are nets with the same sets of input and output channels; $N_1$ is the redex of $R$. If $N_1$ is a subnet of $N$ then $N$ can be reduced by the rule $R$. The result is the net obtained by replacing an occurrence of $N_1$ in $N$ by $N_2$.

The reduction rules are partitioned into two sets. The first set of rules deals with nets constructed over arbitrary (primitive and non-primitive) nodes. In the second set of rules, the redex always contains primitive nodes.

Here is the first set of reduction rules.

DEFINITION 2 (Reductions for Arbitrary Nodes).

$R_1$. Remove an isolated subnet.

$R_2$. Replace a terminating subnet with $k$ input channels by the terminating node with $k$ channels ($k > 0$).

Figures 5 and 6 list the reductions for primitive nodes.

The reductions $S_1$–$S_5$ and $S_7$–$S_{10}$ are self explanatory. Strictly speaking, these are rule schemes. For example, in $S_7$, the redex is a net consisting of $n$-plicator, a terminating node with $m$ ports, and $k$ channels from $n$-plicator to the terminating node. The result of the reduction is the net consisting of an $(n-k)$-plicator, a terminating node with $m-k$ ports (if $m-k>0$), and no channels between these nodes. If $m-k=0$ then the result of the reduction is the net consisting of $(n-k)$-plicator.

In the figure for the rule $S_7$, as well as in the figures for the rules $S_2$–$S_6$, we used an abbreviated syntax and did not point explicitly to the number of channels for the multi-plicator and the terminating nodes. In the rules $S_8$–$S_{10}$ we use 1-plicators and their labels appear in the figures. Note also that in $S_8$ the redex can be $Copy(E)$ node, where $E$ is any restricted regular expression that denotes $\varepsilon$ stream; all such expressions are constructed from $\varepsilon$ by concatenation and iteration.

Let us comment on rule $S_6$. In the reduction scheme $S_6$ the redex is a cyclic net which contains only *Copy* nodes and $n$-plicator nodes. The result of the reduction is a set of generators. Every generator corresponds to an exactly one output channel of the redex net. The generator which corresponds to an output channel *ch* of the redex periodically produces the stream $s$ which is constructed as follows: (1) traverse the cycle starting from *ch* and go in the direction opposite to the channel direction; (2) concatenate all parameters of the Copy nodes on this path. For example, in Fig. 5, $x=(sut)^{\omega}$, $y=(tsu)^{\omega}$, $z=(uts)^{\omega}$. Because of this rule, we use restricted regular expressions as notations for the parameters of generator nodes.

Note that rules $R_1$, $R_2$, and $S_6$ are of a different nature from that of the other rules, in the sense that they apply to arbitrary big subnets as redexes. The other rules are local and their redexes contain only two nodes.

## *3.3. Normal Form Theorem*

We say that a net is in a *normal form* if it does not contain redexes. A net $N'$ in the normal form is called a normal form of $N$ if $N$ is reducible to $N'$ by a sequence of reductions.

THEOREM 1. *Every net has a normal form. There exists a polynomial algorithm which constructs a normal form of a net.*

In the rest of this section a proof of Theorem 1 is presented.

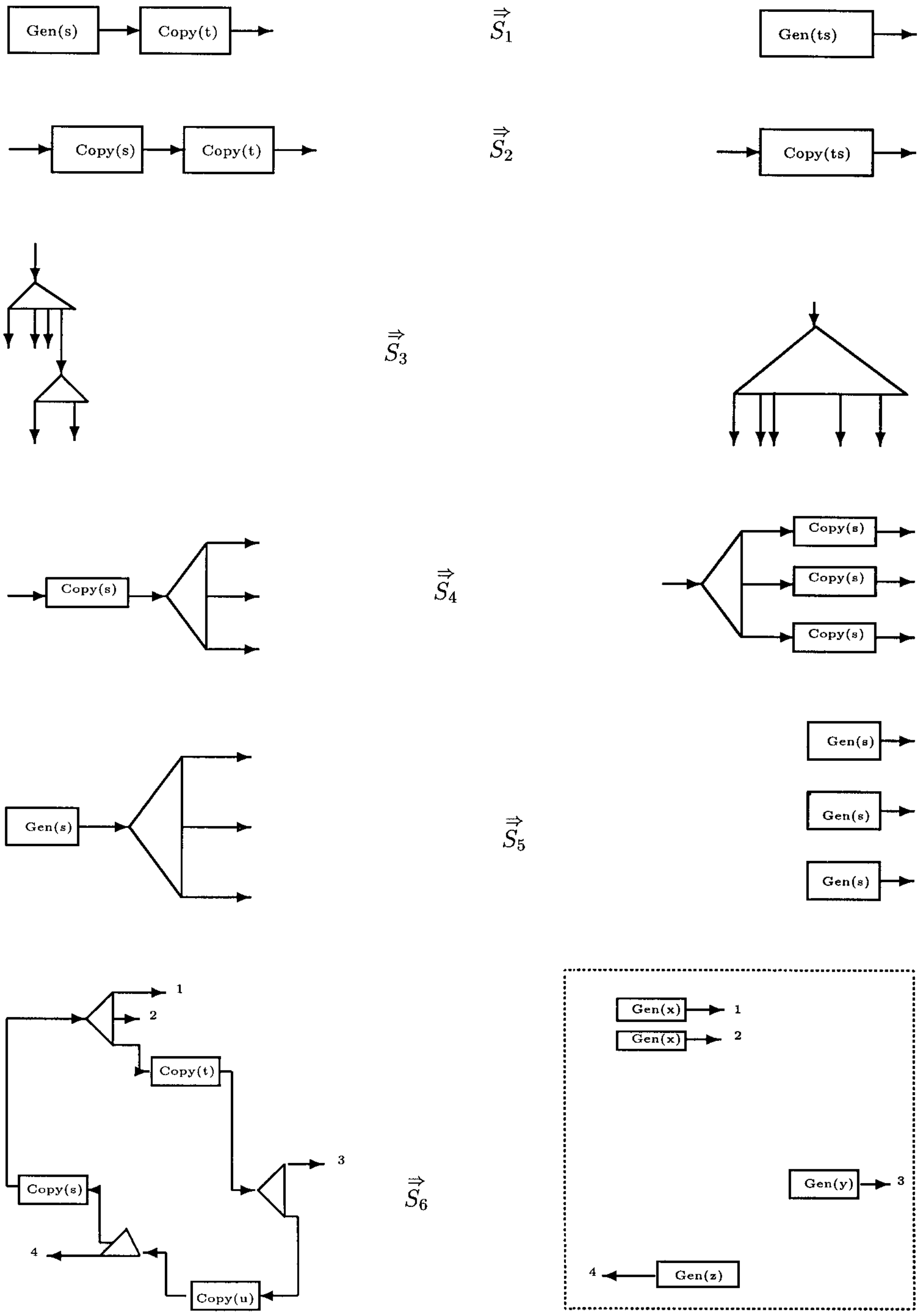
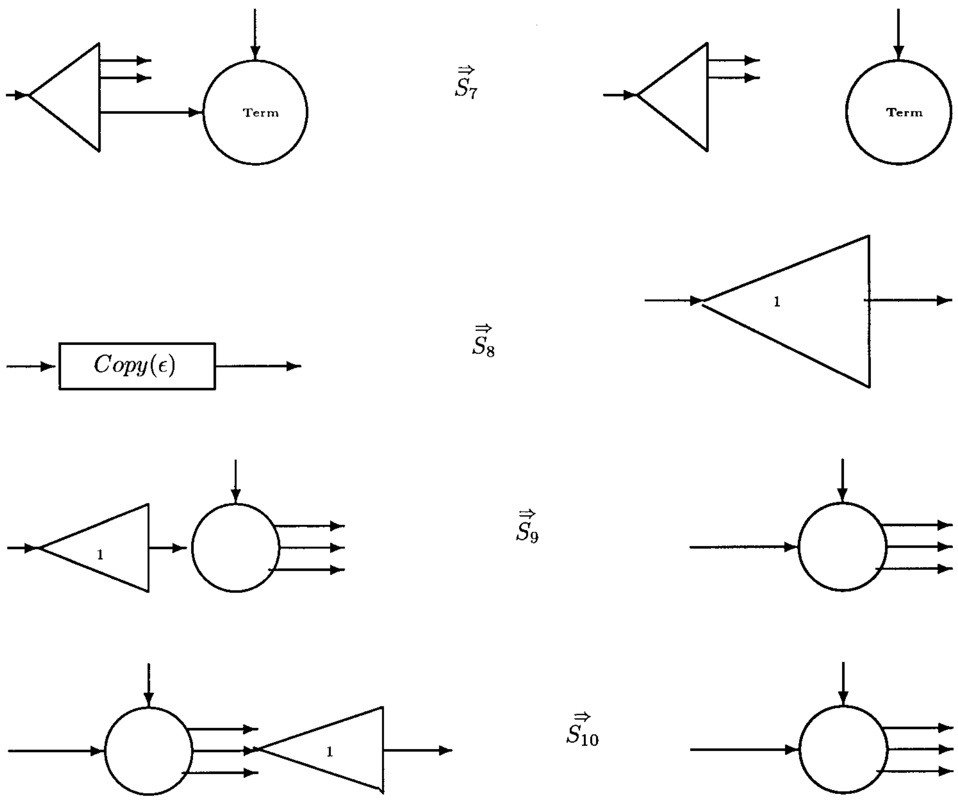

Gen(s)
Copy(t)
$\overset{\Rightarrow}{S}_1$
Gen(ts)
Copy(s)
Copy(t)
$\overset{\Rightarrow}{S}_2$
Copy(ts)
$\overset{\Rightarrow}{S}_3$
Copy(s)
$\overset{\Rightarrow}{S}_4$
Copy(s)
Copy(s)
Copy(s)
Gen(s)
$\overset{\Rightarrow}{S}_5$
Gen(s)
Gen(s)
Gen(s)
1
2
Copy(t)
3
Copy(s)
4
Copy(u)
$\overset{\Rightarrow}{S}_6$
Gen(x)
1
Gen(x)
2
Gen(y)
3
4
Gen(z)

**FIG. 5.** Reductions for primitive nodes.



Term
$\overset{\Rightarrow}{S_7}$
Term
$Copy(\epsilon)$
$\overset{\Rightarrow}{S_8}$
1
1
$\overset{\Rightarrow}{S_9}$
1
$\overset{\Rightarrow}{S_{10}}$

**FIG. 6.** Reductions for primitive nodes.

*Proof.* Let us partition the reduction rules into five groups $G_1, ..., G_5$ as follows:

$G_1$: rules $R_1$ and $R_2$.

$G_2$: rules $S_6$ and $S_7$.

$G_3$: rule $S_8$.

$G_4$: rule $S_4$.

$G_5$: rules $S_1$, $S_2$, $S_3$, $S_5$, $S_9$, and $S_{10}$.

We say that $N$ is in $G_i$ normal form if it contains no redex for the rules in $G_i$. We say that $N$ is in $G_{<i}$ normal form if it contains no redex for the rules in $\bigcup \{G_j : j < i\}$. We say that $N$ is in $G_{\leqslant i}$ normal form if it contains no redex for the rules in $\bigcup \{G_j : j \leqslant i\}$.

We say that $N$ is $G_i$ normal form of $N'$ if $N$ is in $G_i$ normal form and $N$ is obtained from $N'$ by a sequence of reductions from the group $G_i$. The notions of $G_{<i}$ and $G_{\leqslant i}$ normal forms of a net are defined in a similar way.

One can check that when a rule $R$ from any one of these groups $G_i$ is applied to a net in $G_{<i}$ normal form, then the result is still in $G_{<i}$ normal form. Therefore,

LEMMA 2. *If $N$ is in $G_{<i}$ normal form then a $G_i$ normal form of $N$ is $G_{\leqslant i}$ normal form of $N$.*

The next five propositions show that $G_i$ normal form can be constructed in polynomial time.

PROPOSITION 3. *The $G_1$ normal form of a net can be constructed in polynomial time.*

*Proof.* The proof of this proposition is precisely the same as the proof of Proposition 1 in [18]. (This proof is given in Apendix A of [18].) ∎

PROPOSITION 4. *The $G_2$ normal form of a net can be constructed in polynomial time.*

*Proof.* Let the $G_2$-size of a net be defined as the number of its channels plus the sum of the length of the parameters of its copy nodes. It is clear that the $G_2$-size of a net $N$ is less than the size of $N$. We show that the $G_2$ normal form of a net $N$ can be found in time polynomial in the $G_2$-size of $N$.

Group $G_2$ contains rules $S_6$ and $S_7$. Observe that there exists a polynomial (in $G_2$-size) algorithm which checks whether a net contains a redex for these rules, and if such a redex exists, the algorithm returns the result of the reduction. This observation is immediate for $S_7$ reductions. $S_6$ redexes can be found in polynomial time because they contain only nodes with one input port. All the paths containing only the nodes with one input port can be found in polynomial time.

Also note that $S_6$ and $S_7$ reductions decrease the number of channels of a net and its $G_2$-size. Therefore, the normal form can be constructed in polynomial time. ∎

PROPOSITION 5. *The $G_3$ normal form of a net can be constructed in polynomial time.*

*Proof.* A $G_3$ redex can be found in polynomial time and it can be reduced in constant time. Every $G_3$ reduction decreases the number of *Copy*$(\varepsilon)$ nodes and does not increase the size of a net. Therefore, the $G_3$ normal form can be constructed in polynomial time. ∎

PROPOSITION 6. *Let $N$ be a net in $G_{<4}$ normal form. The $G_4$ normal form of $N$ can be constructed in polynomial time.*

*Proof.* $S_4$ is the only rule in $G_4$.

Note that $S_4$ affects only the subnet consisting of *Copy* and multi-plicator nodes. Such a subnet can be found in polynomial time. Moreover, if $N'$ is a net in $G_{<4}$ normal form then, the subnet of $N'$ consisting of all its copy and generator nodes is in $G_{<4}$ normal form.

Therefore, it is enough to show that a $G_4$ normal form can be constructed in polynomial time for any net $N$ which is in $G_{<4}$ normal form and contains only copy and multiplicator nodes.

Such a net $N$ should be acyclic, because the only possible cycles of *Copy* and multi-plicator nodes are redexes of rules $R_1$, $R_2$ or $S_6$ (and these rules are in groups $G_1$ and $G_2$).

Let $N$ be an acyclic net consisting of *Copy* and multi-plicator nodes. From every node $n$ of the net there is a path to an output channel of $N$. We define the weight of a path from a node to an output channel as the number of multi-plicator nodes on the path. We define the weight of a node as the sum of the weights of all the paths from the node to all output ports. We define the weight of a net as the sum of the weights of its *Copy* nodes plus the number of its nodes.

Let us show that the weight of an $n$ nodes acyclic net which contains only *Copy* and multi-plicator nodes is bounded by $2n^3$. Indeed, Copy and multi-plicator nodes have only one input channel and therefore any acyclic net over such nodes has the form of a forest (set of trees). The output channels are the leaves of this forest. In any forest over $n$ nodes the number of all paths to all leaves is bounded by $n^2$. Note that the weight of any path is bounded by $n$. Hence, the weight of the net is bounded by $n \times n^2 + n \leqslant 2n^3$.

Recall that the size of a net $N$ is the number of its nodes plus the number of its channels plus the length of the expressions which appear as parameters of the copy and generator nodes of $N$. Note that the size of a net is bounded by its number of nodes multiplied by its degree (degree of a net is the maximal number of ports for the nodes in the net) multiplied by MAX-EXP (MAX-EXP is the length of the maximal expression which appears as a parameter of its copy and generator nodes), and it is less than the weight of the net multiplied by its degree multiplied by MAX-EXP.

Finally,

(1) An application of $S_4$ reduces the weight. Therefore, only finite sequences of $S_4$ reductions can be applied to $N$ and the length of such a sequence is bounded by the weight of $N$.

(2) Note that the size of all the nets in a reduction sequence is bounded by the weight of $N$ multiplied by its degree multiplied by MAX-EXP.

(3) It is clear that an $S_4$ redex can be found in polynomial time and $S_4$ reduction can be performed in polynomial time.

From (1), (2), (3) and the fact that the weight of a net is bounded by $2n^3$, it follows that $G_4$ normal form can be constructed in polynomial time. ∎

PROPOSITION 7. *The $G_5$ normal form of a net can be constructed in polynomial time.*

*Proof.* A $G_5$ redex can be found in polynomial (in number of nodes + number of channels) time and it can be reduced in linear time. Every $G_5$ reduction decreases the number of nodes + number of channels of a net. Therefore, $G_5$ normal form can be constructed in polynomial time. ∎

Now we are ready to prove Theorem 1. Let $N_0$ be a net. Let $N_1$ be its $G_1$ normal form and let $N_i$ be the $G_i$ normal form of $N_{i-1}$, $i=2, 3, 4, 5$.

By Lemma 2, $N_5$ is a normal form of $N_0$. By Propositions 3, 4, 5, 6, and 7, $N_i$ can be constructed from $N_{i-1}$ in polynomial time. Therefore, a normal form of a net can be constructed in polynomial time. ∎

## 4. SOUNDNESS

THEOREM 8 (Soundness of the Reductions). *If $N$ is reducible to $N'$ then for all interpretations, $N$ and $N'$ have the same Input–Output behavior.*

*Proof.* Consider any of our reduction rules $Red: N_1 \Rightarrow N_2$. It is easy to check that $N_1$ and $N_2$ have the same I–O behavior under all interpretations. Unfortunately, this fact still does not imply the soundness of our reduction rules.

Indeed, recall that in [2], Brock and Ackerman proved that I-O equivalence is not substitutive. They provided an example of two interpreted nets $N_1$ and $N_2$ and a context $C[\ ]$ such that $N_1$ and $N_2$ are I–O equivalent but $C[N_1]$ and $C[N_2]$ are not I–O equivalent. This fact was such a surprise when it was first discovered that it is known as the Brock–Ackerman anomaly.

Since I–O equivalence is not substitutive, we have to find a substitutive equivalence $\equiv$ which refines I–O equivalence and to show that for any of our reduction rules $Red: N_1 \Rightarrow N_2$ the nets $N_1$ and $N_2$ are not only I–O equivalent, but also are $\equiv$ equivalent.

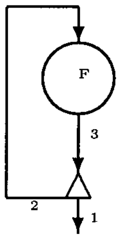
There exist several different formalizations of dataflow semantics [5, 6, 10, 19, 20]. As usually happens, a precise formalization of semantics is based on many definitions and is quite lengthy. (For an informal presentation of the semantics of dataflow nets we refer the reader to Section 3.2 of [18].) This is why we have not presented these definitions here and we state only that the verification of soundness of reduction rules $R_1$, $R_2$, $S_9$, and $S_{10}$ is a routine task for the formalizations referred to above. Actually, the soundness of $R_1$, $R_2$ and $S_9$, $S_{10}$ is a sanity test for a dataflow semantics.

In order to justify the soundness of the rules $S_1$–$S_8$ we recall the Kahn Principle [11].

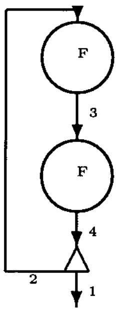
Kahn [11] considered dataflow networks over a class of specific determinate automata. The Input–Output behaviors of Kahn's automata are continuous functions on the stream domain. Moreover, I–O behavior of a net over such automata is also a function. The Kahn principle states that the function computed by a net is obtained as the minimal solution for an appropriate system of equations which is constructed from the functions computed by the net's components and the net topology. Figure 7 illustrates two nets and their corresponding systems of equations. (The primitive node 2-plicator is pictured here as a triangle). The Kahn Principle (KP) and the equality of simultaneous and the corresponding nested least fixed points imply the following

*Property* 1. Let $N$ be a dataflow net consisting of determinate automata and let $N'$ be a subnet of $N$. Let $N''$ be a net consisting of determinate automata which have the same I–O behavior as net $N'$. Then the net obtained from $N$ by replacing $N'$ by $N''$ have the same I–O behavior as $N$.

We omit the Kahn's description of the determinate nodes. For our proof it is only important that all our primitive nodes are determinate. The Kahn principle holds in all the formalizations of dataflow semantics cited above. We refer the



$$\Sigma_1 = \begin{cases} x_1 &= x_3 \\ x_2 &= x_3 \\ x_3 &= F(x_2) \end{cases}$$



$$\Sigma_2 = \begin{cases} x_1 &= x_4 \\ x_2 &= x_4 \\ x_3 &= F(x_2) \\ x_4 &= F(x_3) \end{cases}$$

**FIG. 7.** System of equations for nets $N_3$ and $N_4$.

reader to [1, 6, 9, 12, 14, 16, 19], where proofs of Kahn's principle and its generalization for different formalizations of dataflow semantics can be found. Moreover, for all these semantics there holds the following generalization of the property 1.

*Property* 2. Let $N$ be an arbitrary dataflow net and let $N'$ be a subnet of $N$ consisting of determinate automata. Let $N''$ be a net consisting of determinate automata which has the same I–O behavior as net $N'$. Then the net obtained from $N$ by replacing $N'$ by $N''$ has the same I–O behavior as $N$.

Hence the nets over determinate automata with the same I–O behavior are replaceable by each other in every context.

All our primitive nodes, Copy, multi-plicators, generators, and terminating nodes, are examples of determinate automata. The left and right sides of reduction rules $S_1$–$S_8$ have the same I–O behavior. Therefore, if $N$ is reducible to $N'$ by any of these rules then $N$ and $N'$ are also I–O equivalent.

## 5. COMPLETENESS OF THE REDUCTION RULES

In this section we prove the completeness theorem:

THEOREM 9 (Completeness). *Two nets have the same I–O behavior under all dataflow interpretations iff they are reducible to isomorphic nets by the reduction rules* $R_1$, $R_2$ *and* $S_1$–$S_{10}$.

The *if* direction follows from the soundness theorem.

We concentrate here on the proof of *only if* direction of the completeness theorem.

In Subsection 5.1, we introduce notions which play an important role in our proof. Subsection 5.2 provides the proof of completeness theorem. Proofs of Propositions 12 and 13 are deferred to the appendixes. Some lemmas in the appendix follow immediately from the definitions and we have omitted their proofs.

## *5.1. Paths, Chains, and Strings*

In this section we recall the notion of a path from [18] and provide a definition of a chain which conservatively extends the definition of a chain for the uninterpreted nets. We also introduce here the notion of string, which was not needed in [18].

DEFINITION 3. A path of a net $N$ is a sequence of one of the following forms

(A) $ap_1v_1q_1p_2v_2q_2 \cdots p_iv_iq_i \cdots q_{n-1}p_nv_n$

(B) $ap_1v_1q_1p_2v_2q_2 \cdots p_iv_iq_i \cdots q_{n-1}p_nv_nq_nb$

where

- $v_1, v_2 \cdots v_n$ are nodes.
- $p_i$ is a number assigned to an output port of $v_i$
- $q_i$ is a number assigned to an input port of $v_i$
- there is a channel from the port $p_{i+1}$ of $v_{i+1}$ to the port $q_i$ of $v_i$
- $a$ is the label of the output channel of $N$ which exits port $p_1$ of $v_1$
- for a sequence of the form (B), $b$ is the label of the input channel entering port $q_n$ of $v_n$.

We say that a path of the form (A) leads to node $v_n$. We say that paths meet in $N$ if they lead to the same node in $N$. We say that a path of the form (B) leads to the input channel $b$.

DEFINITION 4. A path is simple if all its nodes are distinct.

DEFINITION 5. A path $s = ap_1v_1q_1p_2v_2q_2 \cdots p_nv_n$ is loop-ended if

1. $v_1 \cdots v_{n-1}$ are different nodes of the nets. and
2. for some $k < n$ the node $v_k$ is the same as $v_n$.

$s$ is loop-ended of index $i$ if among $v_1 \cdots v_{k-1}$ there are exactly $i-1$ non-primitive nodes.

DEFINITION 6. A proper path is a simple or a loop-ended path leading to a generator, to a non-primitive node, or to an input channel.

A path represents a way of traversing a net starting from an output channel.

DEFINITION 7. A chain is obtained from a path by replacing the nodes by their labels and then deleting all multi-plicators and their ports.

*Remark.* This definition of a chain conservatively extends the definition of a chain given in [18]. There, only the nets over the uninterpreted nodes were considered and chains were obtained from a path only by replacing the nodes by their labels.

A chain $s$ is a simple chain in net $N$ if it was obtained from a simple path of $N$. Note that the same chain $s$ can be obtained from several different paths. However, a chain is simple if at least one of these paths is simple. A chain is a loop-ended chain of index $i$ in net $N$ if it was obtained from a loop-ended path of index $i$. A chain leads to a node $v$ if it was obtained from a path which leads to the node $v$. Chains meet if they were obtained from paths which meet. A chain leads to an input port $b$ if it was obtained from a path which leads to the input port $b$. A chain is a proper chain in net $N$ if it was obtained from a proper path of $N$.

Example. Chain $41K23L12M11$ is a simple chain of the net in Fig. 4. Chain $41K23L22M14K$ is a loop-ended chain of index 1 in the net in Fig. 4.

Definition 8. A string is obtained from a chain by deleting copy labels and their ports.

A string is simple (proper, loop-ended of index $i$) in $N$ if it is obtained from a simple (proper, loop-ended of index $i$) chain in $N$.

Lemma 10. *For every net in normal form there exists at most one path which corresponds to a chain.*

*Proof.* The proof is by induction on the length of the chain. It follows from: (1) Paths and chains start at output channels of a net. (2) The output channels have different labels. (3) A port of a node is connected to exactly one channel. (4) The ports of a non-primitive node have different numbers. (5) Multi-plicators have only one input port. (6) In a net in normal form a multi-plicator is never connected to a multi-plicator (reduction rule $S_3$). ▮

Lemma 11. *For every net in normal form there exists at most one proper path which corresponds to a string.*

We say that proper strings meet in $N$ if their corresponding proper paths meet in $N$.

## 5.2. Proof of Completeness Theorem

The proof is based on the following propositions:

Proposition 12. *Let $N_1$ and $N_2$ be nets in normal form with the same external channels. $N_1$ is isomorphic to $N_2$ iff*

1. *$N_1$ and $N_2$ have the same set of simple proper chains.*
2. *$N_1$ and $N_2$ have the same set of loop-ended chains of index i.*
3. *Simple proper strings meet in $N_1$ iff they meet in $N_2$.*

*Proof.* See appendix A. ▮

Proposition 13. *Let $N_1$ and $N_2$ be nets in normal form which are I–O equivalent under all interpretations. Then*

1. *$N_1$ and $N_2$ have the same set of simple proper chains.*
2. *$N_1$ and $N_2$ have the same set of loop-ended chains of index i.*
3. *Simple proper strings meet in $N_1$ iff they meet in $N_2$.*

*Proof.* See appendix B. ▌

The completeness proof proceeds as follows. Assume that nets $N_1$ and $N_2$ are I–O equivalent under all interpretations. Let $N'_1$ and $N'_2$ be normal forms of nets $N_1$ and $N_2$ (they exist by Proposition 1). The soundness theorem implies that $N'_1$ and $N'_2$ are also I–O equivalent under all interpretations. Therefore, by Propositions 13 and 12, nets $N'_1$ and $N'_2$ are isomorphic. Hence, $N_1$ and $N_2$ are reducible to isomorphic nets.

Also note that the soundness theorem, Theorem 1, and Propositions 12 and 13 imply

THEOREM 14. (Uniqueness of Normal Form). *Every net has a unique (up to isomorphism) normal form.*

## 6. COMPLEXITY OF STRUCTURAL EQUIVALENCE

The main result of this section is:

THEOREM 15. *I–O equivalence of partially interpreted nets under all interpretations is decidable in polynomial time.*

*Proof.* Theorem 15 follows from the completeness theorem, Theorem 14, Theorem 1, and Proposition 16 below. ▌

PROPOSITION 16. *There exists a polynomial algorithm which checks whether two nets in normal form are isomorphic.*

Proposition 16 follows from Proposition 17 and Proposition 19 given below.

PROPOSITION 17. *There exists a polynomial algorithm which checks whether a map $\phi$ between the nodes of two nets is an isomorphism.*

*Proof.* To verify whether $\phi$ is an isomorphism we have to check conditions 1, 2, and 3 of Definition 1 (Section 2.3). The only non-trivial part is to check that $\phi$ preserves the labeling of nodes, i.e., $v$ and $\phi(v)$ have the same label.

If $v$ is an uninterpreted node, a terminating node, or a multi-plicator this condition is easily verified. Otherwise, $v$ is labeled by *Gen*($e$) or *Copy*($e$), where $e$ is a restricted regular expression (see Section 2.1).

Therefore, Proposition 17 follows from

LEMMA 18. *Equality between restricted regular expressions is decidable in polynomial time.*

*Proof.* Recall that expressions are equal if they denote the same stream.

For every restricted regular expression we can find in polynomial time an equal expression in one of the following forms:

*Form A.* $a_1 \cdots a_n$

*Form B.* $a_1 \cdots a_n(b_0 \cdots b_m)^\omega$, where $a_n \neq b_m$ and $b_0 \cdots b_m$ is not periodic (i.e., $m = 0$ or $b_0 \cdots b_m \neq (b_0 \cdots b_{m/k})^k$ for every $k > 1$).

Clearly, the equality between expressions in the form (A) can be checked in linear time. An expression in the form (A) is never equal to an expression in the form (B). Expressions $a_1 \cdots a_n(b_0 \cdots b_m)^\omega$ and $a'_1 \cdots a'_{n'}(b'_0 \cdots b'_{m'})^\omega$ in the form (B) are equal if and only $n = n' \wedge m = m'$ and $\forall i \leqslant n . a_i = a'_i \wedge \forall j \leqslant m . b_j = b'_j$. Hence, the equality between restricted regular expressions can be tested in polynomial time. ▮

PROPOSITION 19. *There exists a polynomial algorithm which for any pair of nets $N_1$ and $N_2$ constructs a relation $R$ between their nodes and the algorithm satisfies the following correctness criteria: If nets $N_1$ and $N_2$ are in normal form and are isomorphic, then $R$ is the graph of an isomorphism between these nets.*

*Proof.* Such an algorithm is described in appendix C of [18]. Actually a stronger result was shown there, namely: if nets $N_1$ and $N_2$ do not contain $R_1$ and $R_2$ redexes and are isomorphic then the algorithm returns a graph of an isomorphism between $N_1$ and $N_2$. ▮

## APPENDIX A: PROOF OF PROPOSITION 12

The proof of Proposition 12 is organized as follows. Subsection A.1 lists some simple properties of the nets in normal form. Subsection A.2 characterizes isomorphic nets in terms of their simple and loop-ended chains. Subsection A.3 reduces Proposition 12 to the propositions of Subsection A.2.

### A.1. Basic Properties of the Nets in Normal Form

In this subsection many lemmas (to which we will refer later) that explore properties of chains for nets in normal form are listed. In most cases their proofs are omitted because they directly follow from the definitions and the reduction rules.

LEMMA 20. *Let $N$ be a net in normal form and let $s$ be a chain of $N$.*

1. *$s$ leads to a multi-plicator iff it has the form $a\, p_1 A_1 q_1 p_2 A_2 q_2 \cdots p_m A_m q_m$.*

2. *$s$ leads to a channel $b$ of the net iff it has the form $a\, p_1 A_1 q_1 p_2 A_2 q_2 \cdots p_m A_m q_m b$.*

3. *$s$ leads to a node which is not labeled by a multi-plicator iff $s$ has the form $a\, p_1 A_1 q_1 p_2 A_2 q_2 \cdots p_m A_m$; if $s$ leads to a node $v$, then $v$ is labeled by $A_m$.*

LEMMA 21. *Let $N$ be a net in normal form and let $a p_1 A_1 q_1 \cdots p_i v_i q_i p_{i+1} v_{i+1} \cdots$ be a path of $N$.*

1. *If $v_i$ is labeled by a multi-plicator, then $v_{i+1}$ is labeled by a non-primitive label.*

2. *If $v_i$ is labeled by a Copy, then $v_{i+1}$ is either a multi-plicator or a non-primitive node.*

LEMMA 22. *Let $N$ be a net in normal form.*

1. *$N$ has at most one terminating node.*

2. *There exists a simple chain leading to a node $v$ of $N$ iff $v$ is not the terminating node.*

3. *There exists no simple chain leading to an input channel $b$ of $N$ iff $b$ enters a port of the terminating node.*

*Proof.* The lemma follows from the proof of Proposition 3. ▌

LEMMA 23. *Let $N$ be a net in normal form, let $v$ be its node, and let $r$ be an output port of $v$. Then exactly one of the following alternative holds:*

1. *There exists a simple or a loop-ended chain which leads to $v$ through $r$.*

2. *The channel exiting port $r$ of $v$ is connected to a port of the terminating node of $N$.*

*Proof.* The lemma follows from the proof of Proposition 3. ▌

LEMMA 24. *Let $q$ be an input port of $v$ and let $h$ be a simple path leading to $v$. Then, if there exists a channel from an output port $p$ of $v'$ to the port $q$ of $v$ then $hqpv'$ is a simple or a loop-ended path of $N$.*

LEMMA 25. Let *$N$ be a net in normal form and let $s$ be its loop-ended chain of index $i$. Then the chain Simplify$(S, i)$ defined in Fig.* 8 *is a simple chain which meets with $s$.*

LEMMA 26. *Let $N$ be a net in normal form. Let $s = ap_1A_1q_1 \cdots p_nA_nq_n$ be a simple or a loop-ended chain of $N$, and let $s' = a'p'_1A'_1q'_1 \cdots p'_mA'_mq'_m$ be a simple or a loop-ended chain of $N$. If $s$ and $s'$ lead to a node $v$ of $N$ through different ports, then $q_n \neq q'_m$ or simple chains $ap_1A_1q_1 \cdots p_nA_n$ and $a'p'_1A'_1q'_1 \cdots p'_mA'_m$ do not meet in $N$.*

*Proof.* Both chains $s$ and $s'$ lead to the node $v$ which is a multi-plicator node by Lemma 20(1). By Lemma 21(1) the node $u$ that precedes $v$ on the path that corresponds to the chain $s$ (this path is unique by Lemma 10) is non-primitive. Therefore $ap_1A_1q_1 \cdots p_nA_n$ leads to $u$. Moreover, $ap_1A_1q_1 \cdots p_nA_n$ is a simple chain because it is a proper prefix of a simple or a loop ended chain. Similarly, $a'p'_1A'_1q'_1 \cdots p'_mA'_m$ leads to $u'$ which precedes $v$ on the path that corresponds to the chain $s'$. Assume that $u = u'$. Since $u$ is a non-primitive node its output ports have different labels and therefore if $q_n = q'_m$ we obtain that $s$ and $s'$ leads to $v$ through the channel which exit the port $q_n$ of $u$ and this contradicts the assumption that $s$

```
Function Simplify($s, i$)
If $s = ap_1A_1q_1 \ldots p_nA_n$ then return the maximal prefix of $s$ of the form
    $ap_1A_1q_1 \ldots p_nA_m$ which contains $i$ non-primitive labels.
If $s = ap_1A_1q_1 \ldots p_nA_nq_n$ then return the maximal prefix of $s$ of the form
    $ap_1A_1q_1 \ldots p_nA_mq_m$ which contains $i - 1$ non-primitive labels.
end
```

**FIG. 8.** Function Simplify($s$, $i$).

and $s'$ lead to $v$ through different ports. Therefore, $q_n \neq q'_m$ or $ap_1A_1q_1 \cdots p_nA_n$ and $a'p'_1A'_1q'_1 \cdots p'_mA'_m$ do not meet in $N$. ▮

LEMMA 27. *Let $N$ be a net in normal form and let $s$ be a simple or a loop-ended chain leading to a node $v$. Moreover, assume that $v$ is not a multi-plicator and that it is labeled by $A$. Then $s$ leads to $v$ through port $p$ iff $s$ has the form $tpA$.*

LEMMA 28. *Let $N$ be a net in normal form.*

1. *There exists a channel from a multi-plicator node $v$ to a port $q$ of a node $v'$ iff there exists (a simple or a loop-ended) chain $s = ap_1A_1q_1 \cdots p_nA_nq_n$ leading to $v$ such that $q_n = q$ and chain $ap_1A_1q_1 \cdots p_nA_n$ leads to $v'$.*

2. *There exists a channel from a port $p$ of node $v$ to the input port of a multi-plicator node $v'$ iff there exists (a simple or a loop-ended) chain $s = ap_1A_1q_1 \cdots p_{n-1}A_{n-1}q_{n-1}p_nA_n$ leading to $v$ such that $p_n = p$ and chain $ap_1A_1q_1 \cdots p_{n-1}A_{n-1}q_{n-1}$ leads to $v'$.*

3. *Port $p$ of a node $v$ is connected to a port of the terminating node iff $p_n \neq p$ for any (a simple or a loop-ended) chain $ap_1A_1q_1 \cdots p_nA_n$ leading to $v$.*

4. *If $v$ and $v'$ are neither a multi-plicator nor a terminating nodes of $N$ then there exists channel from port $p$ of $v$ to port $q$ of $v'$ iff there exists (a simple or a loop-ended) chain $s = ap_1A_1q_1 \cdots p_{n-1}A_{n-1}q_{n-1}p_nA_n$ leading to $v$ such that $p_n = p$, $q_{n-1} = q$, chain $ap_1A_1q_1 \cdots p_{n-1}A_{n-1}$ leads to $v'$ and $ap_1A_1q_1 \cdots p_{n-1}A_{n-1}q_{n-1}$ is not a chain of $N$.*

LEMMA 29 (External Channels). *Let $N$ be a net in normal form.*

1. *An output channel $a_1$ of $N$ exits an output port of a multi-plicator node $v$ iff the chain $a_1$ leads to $v$ in $N$.*

2. *Let $v$ be a node of $N$ labeled by $A$ and $A$ is not a multi-plicator. An output channel $a_1$ of $N$ exits an output port $p$ of $v$ iff $a_1pA$ is a simple chain in $N$ which leads to $v$ and $a_1$ is not a chain of $N$.*

3. *Let $v$ be the terminating node of $N$. An input channel $b$ of $N$ enters an input port of $v$ iff no simple chain in $N$ leads to $b$.*

4. *Let $v$ be a multi-plicator node. An input channel $b$ of $N$ enters the input port of $v$ iff there exists a simple chain $ap_1A_1q_1 \cdots p_{n-1}A_{n-1}q_{n-1}b$ in $N$ such that $ap_1A_1q_1 \cdots p_{n-1}A_{n-1}q_{n-1}$ leads to $v$.*

5. *Let $v$ be neither a multi-plicator nor a terminating node of $N$. An input port $b$ of $N$ enters an input port $q$ of $v$ iff there exists a simple chain $ap_1A_1q_1 \cdots p_{n-1}A_{n-1}q_{n-1}p_nA_n$ leading to $v$ such that $ap_1A_1q_1 \cdots p_{n-1}A_{n-1}q_{n-1}p_nA_nqb$ is a simple chain of $N$ and $ap_1A_1q_1 \cdots p_{n-1}A_{n-1}q_{n-1}p_nA_nq$ is not a chain of $N$.*

## A.2. First Isomorphism Theorem

In this section we prove the following proposition which characterizes isomorphic nets.

PROPOSITION 30 (First Isomorphism Theorem). *Let $N_1$ and $N_2$ be nets in normal form with the same external channels. $N_1$ is isomorphic to $N_2$ iff*

1. *$N_1$ and $N_2$ have the same set of simple chains.*
2. *$N_1$ and $N_2$ have the same set of loop-ended chains of index i.*
3. *Simple chains meet in $N_1$ iff they meet in $N_2$.*

The *only if* direction of the proposition is trivial. Below we prove the *if* direction.

Define the relation $R$ between the nodes of $N_1$ and $N_2$ as follows:

$v_1 R v_2$ iff either both nodes are terminating nodes or there exists a simple chain leading to $v_1$ in $N_1$ and to $v_2$ in $N_2$.

We claim that $R$ is an isomorphism between the nets.

Subsection A.2.2 provides a proof that $R$ is a graph of an isomorphism; this proof is based on lemmas given in Subsection A.2.1.

### *A.2.1. Some Facts about R*

LEMMA 31. *Assume that $v_1 R v_2$.*

1. *A simple chain s leads to $v_1$ in $N_1$ iff it leads to $v_2$ in $N_2$.*
2. *A loop-ended chain s leads to $v_1$ in $N_1$ iff it leads to $v_2$ in $N_2$.*

*Proof.* (1) Since $v_1 R v_2$ there exists a simple chain $s'$ which leads to $v_1$ in $N_1$ and to $v_2$ in $N_2$. By the assumption $s$ and $s'$ meet in $N_1$ iff they meet in $N_2$. Therefore, $s$ leads to $v_1$ in $N_1$ iff it leads to $v_2$ in $N_2$.

(2) Let $s$ be a loop-ended chain of index $i$ in $N_1$. By the assumption (2) of Proposition 30, $s$ is a loop-ended chain of index $i$ in $N_2$. Let $s_1$ be a simple chain defined as Simplify($s$, $i$) (see Fig. 8). By Lemma 25, chains $s$ and $s_1$ meet both in $N_1$ and in $N_2$.

Since $v_1 R v_2$ there exists a simple chain $s'$ such that it leads to $v_1$ in $N_1$ and to $v_2$ in $N_2$.

By assumption (3) of Proposition 30, simple chains $s_1$ and $s'$ meet in $N_1$ iff they meet in $N_2$. Therefore, $s$ leads to $v_1$ in $N_1$ iff it leads to $v_2$ in $N_2$. ▮

LEMMA 32. *If $v_1 R v_2$ and the label of $v_1$ is neither multi-plicator nor terminating, then $v_1$ and $v_2$ have the same label.*

*Proof.* Follows immediately from Lemma 20(3) and the definition of $R$. ▮

LEMMA 33. *If $v_1 R v_2$ and there is a channel from an output port p of $v_1$ to a port of the terminating node of $N_1$ then there is a channel from port p of $v_2$ to the terminating node of $N_2$.*

*Proof.* $N_1$ is in normal form and $v_1$ is connected to the terminating node of $N_1$. Therefore, by the reduction rule $S_7$, $v_1$ is not a multi-plicator node. By Lemma 32, $v_1$ and $v_2$ have identical labels.

We arrive to a contradiction by assuming that port $p$ of $v_2$ is not connected to the terminating node.

By this assumption and Lemma 23, there is a chain $s$ (simple or loop-ended) leading to $v_2$ through port $p$.

By Lemma 27, $s$ has the form $tpA$.

By Lemma 31, $s$ leads to $v_1$ in $N_1$.

$s$ has the form $tpA$; therefore it should pass through port $p$ of $v_1$. But since this port is connected to the terminating node, $s$ should pass through the terminating node. However, no chain can pass through a terminating node (by Lemma 22(2)). We have arrived at a contradiction. ▮

LEMMA 34. *If $v_1 R v_2$ and $v_1$ is a multi-plicator, then $v_2$ is a multi-plicator.*

*Proof.* Immediately from Lemma 20(1). ▮

LEMMA 35. *Let $v_1 R v_2$ and $v_1$ be a multi-plicator node. If two chains* (*simple or loop-ended*) *lead to $v_1$ through different ports then they lead to $v_2$ through different ports.*

*Proof.* Let $s = ap_1 A_1 q_1 \cdots p_n A_n q_n$ and $s = a' p'_1 A'_1 q'_1 \cdots p'_n A'_n q'_m$ be two chains which lead to $v_1$ in $N_1$ through different ports.

By our assumption $v_1 R v_2$, hence by Lemma 31, chains $s$ and $s'$ lead to $v_2$ in $N_2$.

$s$ and $s'$ lead through different ports to $v_1$; therefore, by Lemma 26, either $q_n \neq q'_m$ or $ap_1 A_1 q_1 \cdots p_n A_n$ and $a' p'_1 A'_1 q'_1 \cdots p'_n A'_m$ do not meet in $N_1$.

If $q_n \neq q'_m$ then these chains lead to different ports of $v_2$ in $N_2$ by Lemma 26.

If $ap_1 A_1 q_1 \cdots p_n A_n$ and $a' p'_1 A'_1 q'_1 \cdots p'_n A'_n q'_m$ do not meet in $N_1$ then they do not meet in $N_2$, because they are simple chains and simple chains meet in $N_1$ iff they meet in $N_2$. Therefore, applying Lemma 26, we obtain that $s$ and $s'$ lead to $v_2$ through different ports. ▮

LEMMA 36. *If $v_1$ is a multi-plicator with $n$ output ports and $v_1 R v_2$ then $v_2$ is a multi-plicator with $n$ output ports.*

*Proof.* By Lemma 34, $v_2$ should be a multi-plicator. Let $m$ be the number of output ports of $v_2$. We are going to show that $n = m$.

By the reduction rule $S_7$, no port of a multi-plicator is connected to a port of a terminating node of a net in normal form.

Therefore, by Lemma 23, there exist $n$ simple or loop-ended chains leading to $v_1$ through different ports. By Lemma 35, these $n$ chains should lead through different ports of $v_2$. Therefore, $v_2$ has at least $n$ output ports. Hence $n \leqslant m$. Symmetrical arguments show that $m \leqslant n$. Therefore, $v_1$ and $v_2$ are multi-plicators with the same number of output channels. ▮

### *A.2.2. R is an Isomorphism*

LEMMA 37. *R is a bijection between the nodes of $N_1$ and $N_2$; i.e.,*

1. (a) *For every node $v_1$ of $N_1$ there exists a node $v_2$ of $N_2$ such that $v_1 R v_2$;* (b) *if $v_1 R v_2$ and $v_1 R v_3$ then $v_2 = v_3$.*

2. (a) *For every node $v_2$ of $N_2$ there exists a node $v_1$ of $N_1$ such that $v_1 R v_2$;* (b) *if $v_1 R v_2$ and $v_3 R v_2$ then $v_1 = v_3$.*

*Proof.* (1) Let $v_1$ be a node of $N_1$. Then $v_1$ is either a terminating node or there is a chain $s$ leading to $v_1$ in $N_1$.

*Case* 1. $v_1$ is not a terminating node

(a) Let $s$ be a simple chain leading to $v_1$ in $N_1$ (such a chain exists by Lemma 22(2)). Then $s$ is a simple chain of $N_2$. Therefore, there exists a node $v_2$ such that $s$ leads to $v_2$ in $N_2$. Hence $v_1 R v_2$, by the definition of $R$. (b) Let $s$ be a simple chain leading to $v_1$. If $v_1 R v_2$ and $v_1 R v_3$ then, by Lemma 31, $s$ leads to $v_2$ in $N_2$ and $s$ leads to $v_3$ in $N_2$. Therefore $v_2 = v_3$.

*Case* 2. $v_1$ is a terminating node of $N_1$.

(a) Let $r$ be a port of $v_1$.

If an input channel $m$ of $N_1$ enters $r$ then, by Lemma 22(3), there is no chain leading to $m$ in $N_1$. Hence, there is no chain leading to $m$ in $N_2$. Applying Lemma 22(3), again we obtain that channel $m$ enters a port of a terminating node of $N_2$. Hence $v_2$ is the terminating node of $N_2$.

If no input channel of $N_1$ enters $r$, then there is a node $v'_1$ in $N_1$ such that its output port is connected to the port $r$ of $v_1$. Node $v'_1$ is not a terminating node; therefore, by case 1, there is a node $v'_2$ in $N_2$ such that $v'_1 R v'_2$. Applying Lemma 33 we obtain that $N_2$ has a terminating node $v_2$. Hence $v_1 R v_2$, by the definition of $R$.

(b) follows from Lemma 22(1).

(2) The proof of (2) is dual to the proof of (1). ▮

LEMMA 38. *R preserves the labeling of nodes.*

*Proof.* Assume that $v_1 R v_2$. We have to show that $v_1$ and $v_2$ have the same label.

If $v_1$ is neither a terminating nor a multi-plicator node then, by Lemma 36, $v_1$ and $v_2$ have the same label.

If $v_1$ is a multi-plicator with $n$ output ports then, by Lemma 36, $v_2$ is a multiplicator with $n$ output ports.

It remains to consider the case where $v_1$ is a terminating node. In this case, $v_2$ is the terminating node of $N_2$ by the definition of $R$. So we have to prove that these nodes have the same number of ports. We are going to show that the number of $v_2$ ports is greater than or equal to the number of $v_1$ ports. Symmetrical arguments demonstrate that the number of $v_1$ ports is greater than or equal to the number of $v_2$ ports. Therefore $v_1$ and $v_2$ have the same number of ports and the same label.

Assume that $v_1$ is a terminating node with $n$ ports. Every port of $v_1$ is connected either to an output port of another node or to an input channel of the net.

Let $b_1 \cdots b_k$ be the input channels of $N_1$ which enter the ports of $v_1$ and let $r_{k+1} \cdots r_n$ be the ports of $v_1$ connected to nodes in $N_1$.

We are going to show that at least $k$ input channels of $N_2$ enter $v_2$ and at least $n-k$ ports of $v_2$ are connected to other nodes in $N_2$. Therefore $v_2$ is a terminating node with at least $n$ ports.

Recall that $b_1 \cdots b_k$ are the input channels of $N_1$ which enter $v_1$. By Lemma 22(3) there is no simple chain in $N_1$ leading to $b_i$, $i = 1, ..., k$. Since $N_1$ and $N_2$ have the same set of simple chains, there is no simple chain in $N_2$ leading to $b_i$, $i = 1, ..., k$. Because $b_i$ are input ports of $N_2$, it follows by Lemma 22(3) that $b_i$ enter the terminating node of $N_2$, i.e., $b_i$ enter $v_2$.

Assume that from port $p_i$ of the node $u_i$ of $N_1$ there is a channel to port $r_i$ of $v_1$, $i=k+1, ..., n$.

All $r_i$ are different ports of the terminating node; therefore

$$\text{for } i \neq j, \text{ either } p_i \neq p_j \text{ or } u_i \neq u_j.$$

By Lemma 37, there are nodes $u'_i$ in $N_2$ such that $u_i R u'_i$ for $i=k+1, ..., n$. Moreover $u_i = u_j$ iff $u'_i = u'_j$.

By Lemma 33, there are channels from port $p_i$ of $u'_i$ to a port of the terminating node of $N_2$.

All these channels are different, because for $i \neq j$ either $p_i \neq p_j$ or $u'_i \neq u'_j$.

Therefore, $n-k$ different ports of $v_2$ are connected to ports of the other nodes in $N_2$. Hence $v_2$ has at least $n$ ports. ▮

Lemma 39. *R preserves the adjacency relation.*

*Proof.* Let $v_1 R v_2$ and $v'_1 R v'_2$. We have to show that there is a channel from port $p$ of $v_1$ to port $q$ of $v'_1$ iff there is a channel from port $p$ of $v_2$ to port $q$ of $v'_2$. Since $N$ is a net in normal form, one of the following cases holds

*Case* 1. $v'_1$ is the terminating node of $N_1$.

*Case* 2. $v_1$ is a multi-plicator.

*Case* 3. $v'_1$ is a multi-plicator.

*Case* 4. $v_1$ and $v'_1$ are neither multi-plicator nor terminating nodes of $N_1$.

We prove each of these cases separately.

*Case* 1. Since $R$ preserves labeling of nodes $v'_2$ is the terminating node of $N_2$.

Assume that there is a channel from port $p$ of $v_1$ to a port of $v'_1$ (all ports of terminating node are labeled by 1).

$v_1$ is not a multi-plicator node by the reduction rule $S_7$. By Lemma 28(3), $p_n \neq p$ for any chain $ap_1A_1q_1 \cdots p_nA_n$ leading to $v_1$ in $N_1$. Therefore, by Lemma 31, $p_n \neq p$ for any chain $ap_1A_1q_1 \cdots p_nA_n$ leading to $v_2$ in $N_2$.

Hence, by Lemma 28(3), port $p$ of $v_2$ is connected to the terminating node $v'_2$ of $N_2$.

*Case* 2. Since $R$ preserves labeling of the nodes $v_2$ is a multi-plicator.

Assume that there is a channel from $v_1$ to a port $q$ of $v'_1$.

By Lemma 28(1), there exists (a simple or a loop-ended) chain $s = ap_1A_1q_1 \cdots p_nA_nq_n$ leading to $v_1$ such that $q_n = q$ and chain $ap_1A_1q_1 \cdots p_nA_n$ leads to $v'_1$.

Therefore, by Lemma 31, $s$ leads to $v_2$ in $N_2$ and $ap_1A_1q_1 \cdots p_nA_n$ leads to $v'_2$ in $N_2$.

Hence, by Lemma 28(1), there is a channel from $v_2$ to the port $q$ of $v'_2$.

We omit the proofs for case 3 and case 4; they are based on Lemma 31 and Lemma 28. ▮

LEMMA 40. $R$ preserves the labeling of external channels; i.e.,

1. *If $v_1 R v_2$ and an output channel $b$ of $N_1$ enters a port $p$ of $v_1$ then the output channel $b$ of $N_2$ enters a port $p$ of $v_2$.*

2. If *$v_1 R v_2$ and an input channel $a_1$ of $N_1$ exits a port $p$ of $v_1$ then the input channel $a_1$ of $N_2$ exits port $p$ of $v_2$.*

*Proof.* Follows immediately from Lemma 29 and Lemma 31. ▮

Finally, Proposition 30 follows from Lemmas 37, 38, 39, and 40.

## A.3. Proof of Proposition 12

First we state Lemmas 41, 42, 43, and 44, which show that the set of simple chains of a net $N$ and the set of pairs of simple chains which meet in $N$ can be found from the set of proper chains of $N$, the sets of loop-ended chains of index $i$ in $N$, and the set of pairs of proper chains which meet in $N$. We omit the proofs of these Lemmas; these proofs are based on the reduction rules and the definition of a chain.

LEMMA 41 (Chains Meet at a Multi-plicator Node). *Let $N$ be a net in normal form. Let $w$ be a chain $ap_1A_1q_1 \cdots A_nq_n$ and let $w'$ be* a *chain of $N$. Moreover, assume that $w$ and $w'$ are simple chains of $N$. Then $w$ and $w'$ meet in $N$ iff one of the following holds*:

1. *There exists a non-primitive label $A$ and its input port $p$ such that $wpA$ and $w'pA$ are proper chains which meet in $N$.*

2. *There exists an input channel $b$ such that $wb$ and $w'b$ are proper chains of $N$.*

LEMMA 42 (Chains Meet at a Copy Node). *Let $N$ be a net in normal form. Let $w$ be a chain $ap_1A_1q_1 \cdots A_nq_n1Copy(t)$ and let $w'$ be a chain of $N$. Moreover, assume that $w$ and $w'$ are simple chains of $N$. Then $w$ and $w'$ meet in $N$ iff one of the following holds*:

1. *There exists a non-primitive label $A$ and its input port $p$ such that $w2pA$ and $w'2pA$ are proper chains which meet in $N$.*

2. *There exists an input channel $b$ such that $w2b$ and $w'2b$ are proper chains of $N$.*

LEMMA 43 (Chains Leading to a Copy Node). *Let $N$ be a net in normal form. Let $w = ap_1A_1q_1 \cdots A_nq_n1Copy(t)$ be a chain. Then $w$ is a simple chain of $N$ iff $N$ has a proper or a loop-ended chain $s$ such that $w$ is a prefix of $s$.*

LEMMA 44 (Chains Leading to a Multi-plicator Node). *Let $N$ be a net in normal form and let $A_n$ be a non-primitive label. Let $w$ be a chain $ap_1A_1q_1 \cdots A_nq_n$ or a chain $ap_1A_1q_1 \cdots A_nq_n1Copy(t)2$ containing $i-1$ non-primitive labels. Then $w$ is a simple chain of $N$ iff at least one of the following holds*:

1. *$n=0$, $w=a$ and $ab$ is a proper chain of $N$.*
2. *$N$ has a loop-ended chain $s$ of index $i$ such that $w$ is a prefix of $s$.*
3. *There exists a port $p$, a non-primitive label $A$ such that $s=wpA$ is a proper chain of $N$ and one of the following sub-cases holds:*

   (a) *There exists a proper chain $s'=a'p'_1A'_1q'_1\cdots A'_mq'_mpA$ in $N$ and proper chains $s$ and $s'$ meet in $N$ and either $q_n \neq q'_m$ or proper chains $ap_1A_1q_1\cdots A_n$ and $a'p'_1A'_1q'_1\cdots A'_m$ do not meet in $N$.*

   (b) *There exists a proper chain $s'=a'p'_1A'_1q'_1\cdots A'_mq'_m1Copy(t')\,2pA$ in $N$ and proper chains $s$ and $s'$ meet in $N$ and either $q_n \neq q'_m$ or proper chains $ap_1A_1q_1\cdots A_n$ and $a'p'_1A'_1q'_1\cdots A'_m$ do not meet in $N$.*

4. *There exists an input port $b$ of $N$ such that $wb$ is a proper chain of $N$ and one of the following sub-cases holds:*

   (c) *There exists a proper chain $s'=a'p'_1A'_1q'_1\cdots A'_mq'_mb$ in $N$ and and either $q_n \neq q'_m$ or proper chains $ap_1A_1q_1\cdots A_n$ and $a'p'_1A'_1q'_1\cdots A'_m$ do not meet in $N$.*

   (d) *There exists a proper chain $s'=a'p'_1A'_1q'_1\cdots A'_mq'_m1Copy(t')2b$ in $N$ and either $q_n \neq q'_m$ or proper chains $ap_1A_1q_1\cdots A_n$ and $a'p'_1A'_1q'_1\cdots A'_m$ do not meet in $N$.*

The next Lemma characterizes isomorphic nets in terms of their proper chains.

LEMMA 45. *Let $N_1$ and $N_2$ be nets in normal form. If*

(A) *$N_1$ and $N_2$ have the same set of simple proper chains, and*

(B) *for every $i$, the nets $N_1$ and $N_2$ have the same set of loop-ended chains of index $i$, and*

(C) *simple proper chains meet in $N_1$ iff they meet in $N_2$,*

*then*

1. *$N_1$ and $N_2$ have the same set of simple chains;*
2. *simple chains meet in $N_1$ iff they meet in $N_2$.*

*Proof.* Let us first show that the assumptions (A), (B), and (C) imply the following two Lemmas.

LEMMA 46. *$s$ is a proper chain of $N_1$ iff it is a proper chain of $N_2$.*

*Proof.* Let $s$ be a proper chain of $N_1$. Then $s$ is either a simple or a loop-ended proper chain of $N_1$.

If $s$ is a simple proper chain of $N_1$ then, by the assumption (A), it is a simple proper chain of $N_2$.

If $s$ is a loop-ended proper chain of $N_1$ then, by the assumption (B), it is a loop-ended chain of $N_2$. Since $s$ leads to an input port, to a non-primitive node or to a generator node it is also a simple proper chain of $N_2$. ∎

LEMMA 47. *Proper chains meet in $N_1$ iff they meet in $N_2$.*

*Proof.* Let $s_1$ and $s_2$ be proper chains which meet in $N_1$. We consider four cases.

*Case* 1. $s_1$ and $s_2$ are simple proper chains of $N_1$. In this case, by the assumption (C) they also meet in $N_2$.

*Case* 2. $s_1$ is a simple proper chain and $s_2$ is a loop-ended proper chain of index $i$ in $N_1$. Let $s_2'$ be the chain *Simplify*$(s_2, i)$ (see Fig. 8). $s_2'$ is a simple proper chain of $N_1$ and it meets with $s_2$, by Lemma 25. Therefore, $s_1$ and $s_2'$ are simple proper chains which meet in $N_1$. Hence, by case 1,

(a) $s_1$ and $s_2'$ meet in $N_2$.

By Lemma 46, $s_1$, $s_2$ and $s_2'$ are proper chains of $N_2$. By Lemma 25, $s_2'$ is a simple proper chain of $N_2$ and $s_2$ and $s_2'$ meet in $N_2$. Therefore, $s_1$ and $s_2$ meet in $N_2$ iff $s_1$ and $s_2'$ meet in $N_2$. By (a) above $s_1$ and $s_2'$ meet in $N_2$. Hence $s_1$ and $s_2$ meet in $N_2$.

*Case* 3. $s_1$ is a a loop-ended proper chain of index $i$ in $N_1$ and $s_2$ is a simple proper chain in $N_1$. This case is dual to case 2.

*Case* 4. $s_1$ is a loop-ended proper chain of index $i$ in $N_1$ and $s_2$ is a loop-ended proper chain of index $j$ in $N_1$. The proof is similar to the proof for case 2. ∎

Recall that Lemmas 41, 42, 43, and 44 state that the set of simple chains of a net $N$ and the set of pairs of simple chains which meet in $N$ can be found from the set of proper chains of $N$, the sets of loop-ended chains of index $i$ in $N$ and the set of pairs of proper chains which meet in $N$. By Lemmas 46 and 47, these sets are the same for $N_1$ and $N_2$. Therefore, $N_1$ and $N_2$ have the same sets of proper chains and proper chains meet in $N_1$ if they meet in $N_2$. This completes our proof of Lemma 45. ∎

Finally, we are ready to prove Proposition 12. Recall that the strings of a net are obtained from the chains of the net by deleting copy labels and their ports. We say that string *st* corresponds to a chain $s$ if *st* is obtained from $s$ by deleting copy labels.

Note also that if $s = ap_1 A_1 q_1 \cdots A_{i-1} q_{i-1} p_i A_i q_i p_{i+1} A_{i+1} q_{i+1} \cdots$ is a chain of a net in normal form and $A_i$ is a copy label then neither $A_{i-1}$, nor $A_{i+1}$ can be a copy label. Therefore,

LEMMA 48. *Let $N$ be a net in normal form. Let $s_1$ and $s_2$ be two proper chains of $N$. Then $s_1$ is equal to $s_2$ iff their corresponding proper strings are equal.*

LEMMA 49. *Let $N$ be a net in normal form. Proper chains meet in $N$ iff their corresponding strings meet in $N$.*

Proposition 12 follows from Lemma 49, Lemma 45, and Proposition 30.

## APPENDIX B. PROOF OF PROPOSITION 13

The structure of this proof is similar to the structure of the proof of Proposition 6 from [18], but this proof is technically more complicated. Below we recall the language for the description of automata which was used in [18] . Example 1 from [18] that represents the essence of construction is also given here. Then several generic examples are provided which motivate the way the proof for the uninterpreted nets [18] should be extended to the proof for partially interpreted nets. Finally, the formal definitions and the proofs are given.

Below automata are described in the language from [18] inspired by the language of Kahn [11].

The command "**read from port** $q$" causes the described automaton to look at its port $q$. If the channel entering $q$ is not empty then the first token from the channel is consumed; otherwise the automaton waits for the channel to become non-empty, and when it happens, the automaton consumes the first token.

The command "**output** $v$ **on port** $p$" appends the value $v$ to the contents of the channel exiting $p$.

In addition to the constructs of Kahn's language, our language contains non-deterministic choice, denoted by $+$.

If $Pr_1$ describes an automaton $A_1$ and $Pr_2$ describes an automaton $A_2$, then $Pr_1 + Pr_2$ describes the automaton which works either as $A_1$ or as $A_2$. Non-deterministic choice is associative and commutative. We use the notation $\sum_{i=k}^{i=n} Pr_i$ for $Pr_k + Pr_{k+1} \cdots Pr_n$ and $\sum_{i=k}^{i=n} \{Pr_i : \phi(i)\}$ for the sum over all $Pr_i$ such that $k \leqslant i \leqslant n$ and the condition $\phi(i)$ holds.

We are going to define an interpretation $Int_s$ which will distinguish the nets containing chain $s$ from all other nets. First, we illustrate our construction by generic examples, and later, the formal definitions are provided.

EXAMPLE 1. Let $s = a2C78B31A45B$ be a chain. It contains four occurrences of non-primitive labels. Therefore, if it is a simple chain of a net $N$ it should pass through four different non-primitive nodes: $v_1$, $v_2$, $v_3$, $v_4$ (see Fig. 9).

We define four automata $At_1$, $At_2$, $At_3$, $At_4$ which correspond to these nodes (see Fig. 9).

$At_4$ corresponds to $v_4$ and it just outputs value $d_3$ on port 5.

$At_3$ corresponds to $v_3$; it reads from port 4 and if the received value is $d_3$, it outputs value $d_2$ on port 1.

$At_2$ corresponds to $v_2$; it reads from port 3 and if the received value is $d_2$, it outputs value $d_1$ on port 8.

$At_1$ corresponds to $v_1$; it reads from port 7 and if the received value is $d_1$, it outputs value $d_0$ on port 2.

The interpretation $Int$ which corresponds to our chain $s$ is defined as follows:

$Int(A) = At_3$ because $A$ is label of $v_3$;

$Int(B) = At_2 + At_4$, because $B$ is the label of $v_2$ and $v_4$;

$Int(C) = At_1$, because $C$ is the label of $v_1$.

The following observation is straightforward:

*Observation* 1. Let $d_0$, $d_1$, $d_2$, $d_3$ be distinct data values. A net without primitive nodes can produce $d_0$ on channel $a$ under the interpretation $Int$ iff $s$ is a simple chain of $N$.

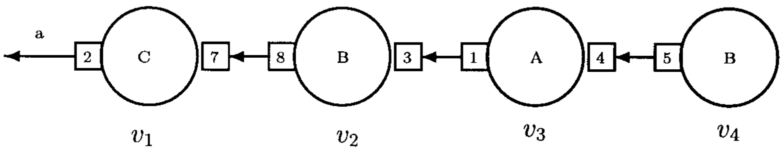


**FIG. 9.** A simple path for chain $s = a2C78B31A45B$.

This example represents the essence of our construction. The general construction is more complicated, due to the presence of copy, generator and multi-plicator nodes. Before describing the general construction we provide several generic examples which illustrate it.

EXAMPLE 2. Let $s = a1\mathit{Copy}(ab)22C78B31\mathit{Copy}(bcb)21A45B$ be a chain. It contains four occurrences of non-primitive labels and two occurrences of *Copy* labels. Therefore, if it is a simple chain of a net $N$, the corresponding path should pass through two copy nodes and four different non-primitive nodes: $v_1$, $v_2$, $v_3$, $v_4$ (see Fig. 10). The path might also contain multi-plicator nodes.

We define four automata $At_1$, $At_2$, $At_3$, $At_4$ which correspond to the non-primitive nodes of the path (see Fig. 10).

$At_4$ corresponds to $v_4$ and it just outputs value $d_3$ on port 5.

$At_3$ corresponds to $v_3$; it reads from port 4 and if the received value is $d_3$, it outputs value $d_2$ on port 1.

$At_2$ corresponds to $v_2$; first it reads from port 3 three incoming values. If these values are not $b$, $c$, $b$ then it stops; otherwise it reads the next value from port 3 and if the received value is $d_2$, it outputs value $d_1$ on port 8.

$At_1$ corresponds to $v_1$; it reads from port 7 and if the received value is $d_1$, it outputs value $d_0$ on port 2.

The interpretation *Int* which corresponds to our chain $s$ is defined as follows:

$$\mathit{Int}(A) = At_3 \text{ because } A \text{ is label of } v_3;$$

$$\mathit{Int}(B) = At_2 + At_4, \text{ because } B \text{ is the label of } v_2 \text{ and } v_4;$$

$$\mathit{Int}(C) = At_1, \text{ because } C \text{ is the label of } v_1.$$

The following observation is straightforward

*Observation* 2. Let $N$ be a net in normal form and assume that $d_0$, $d_1$, $d_2$, $d_3$ are distinct values that do not appear in the parameters of copy and generator nodes of $N$. Such a net $N$ can produce $abd_0$ on channel $a$ under the interpretation *Int* iff $s$ is a simple chain of $N$.

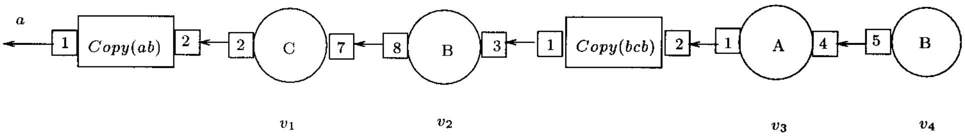


**FIG. 10.** A simple path for chain $s = a1\mathit{Copy}(ab)22C78B31\mathit{Copy}(bcb)21A45B$.

EXAMPLE 3. Let $s = a8B32A45B11Gen(c(ab)^{\omega})$ be a chain. It contains three occurrences of non-primitive labels and one occurrences of *Gen* labels. Therefore, if it is a simple chain of a net *N*, the corresponding path should pass through a generator node and three different non-primitive nodes: $v_1$, $v_2$, $v_3$ (see Fig. 11). The path might also contain multi-plicator nodes.

We define three automata which correspond to the non-primitive nodes of the path (see Fig. 11).

For $v_3$ we define an infinite sequence $\{At_3^{(n)}\}_{n=0}^{\infty}$ of automata; $At_3^{(k)}$ first reads $k$ data values from port 1 and if these $k$ values agree with the prefix of length $k$ of $c(ab)^{\omega}$ then the automaton outputs $d_2$ on port 5. Otherwise it does not produce any output.

$At_2$ corresponds to $v_2$; it reads from port 4 and if the received value is $d_2$, it outputs value $d_1$ on port 2.

$At_1$ corresponds to $v_1$; it reads from port 3 the incoming data value. If the received value is $d_1$, it outputs value $d_0$ on port 8.

We define a sequence of the interpretations $\{Int^{(n)}\}_{n=0}^{\infty}$ which corresponds to our chain, as follows:

$$Int^{(k)}(A) = At_2 \text{ because } A \text{ is the label of } v_2;$$

$$Int^{(k)}(B) = At_1 + At_3^{(k)}, \text{ because } B \text{ is the label of } v_1 \text{ and } v_3.$$

The following observation is straightforward

*Observation* 3. Let *N* be a net in normal form and assume that $d_0$, $d_1$, $d_2$ are distinct values that do not appear in the parameters of copy and generator nodes of *N*. Such a net *N* produces $d_0$ on channel $a$ under all interpretations $Int^{(k)}$ iff $s$ is a simple chain of *N*.

EXAMPLE 4. Let $s = a2C78B31C45B$ be a loop-ended chain of index 2. A corresponding path contains four occurrences $v_1$, $v_2$, $v_3$, $v_4$ of non-primitive nodes. The second occurrence $v_2$ of a non-primitive node coincides with the fourth occurrence $v_4$ of a non-primitive node. A corresponding path is given in Fig. 12. A path for this loop-ended chain also might contain multi-plicator nodes.

We define four automata $At_1$, $At_2$, $At_3$, $At_4$ which correspond to the occurrences of non-primitive nodes.

$At_4$ corresponds to $v_4$ and it just outputs value $d_3$ on port 5.

$At_3$ corresponds to $v_3$; it reads from port 4 and if the received value is $d_3$, it outputs value $d_2$ on port 1.

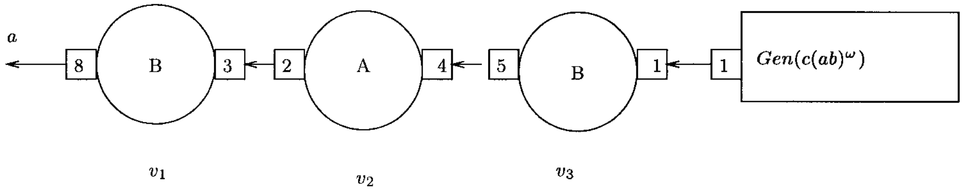


**FIG. 11.** A simple path for chain $s = a8B32A45B11Gen(c(ab)^{\omega})$.

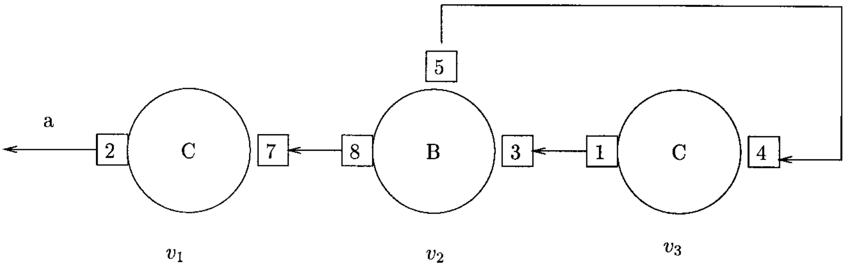


**FIG. 12.** A simple path for chain $s = a2C78B31C45B$.

$At_2$ corresponds to $v_2$; it reads from port 3 and if the received value is $d_2$, it outputs value $d_1$ on port 8.

$At_1$ corresponds to $v_1$; it reads from port 7 and if the received value is $d_1$, it outputs value $d_0$ on port 2.

The interpretation *Int* which corresponds to our loop-ended chain $s$ is defined as follows:

$Int(B) = At_4; At_2$ because $B$ is the label of $v_2$ and $v_4$ and these nodes coincide;

$Int(C) = At_1 + At_3$, because $C$ is the label of $v_1$ and $v_3$.

In this example the interpretation assigns to $B$ the automaton $At_4; At_2$ which first operates like $At_4$ and then like $At_2$.

The following observation is straightforward

*Observation* 4. Let $N$ be a net in normal form and assume that $d_0$, $d_1$, $d_2$, $d_3$ are distinct values that do not appear in the parameters of copy and generator nodes of $N$. Such a net $N$ can produce $d_0$ on channel $a$ under the interpretation *Int* iff $s$ is a loop-ended chain of index 2 of $N$.

*Notations.* The following notations will be used in our proof:
We chose a sequence $d_0, d_1, ...$ of distinct data values; the elements of the sequence will not appear in the parameters of the nets we are testing.

- *Decrement*$(p, q, n)$ describes an automaton which reads a token from channel $q$. If the received token is equal to $d_n$, the automaton will output $d_{n-1}$ on port $p$.

- *Test*$(q, t)$ describes an automaton that tests whether the stream received on port $q$ starts from the finite stream $t$; if so, the automaton successfully stops, otherwise it diverges.

In order to describe the general construction we also introduce the following notations:

Let $s = ap_1 A_1 q_1 \cdots$ be a chain of a net in normal form.

Let $B_1 \cdots B_m$ be the subsequence of non-primitive labels in $s$. We denote by $\overline{p_i}$ (respectively by $\overline{q_i}$) the input (respectively the output) port of $B_i$ on chain $s$.

In the chain $s$, label $B_i$ is either immediately follows $B_{i-1}$ or there exists exactly one label $Copy(t)$ between $B_{i-1}$ and $B_i$ $(i>1)$. We associate with $s$ a sequence $\{t_0, t_1 \cdots t_m\}$ of streams.

$$t_i = \begin{cases} \varepsilon & \text{if } B_{i+1} \text{ precedes immediately by } B_i \text{ and } 1<i<m \\ \varepsilon & \text{if } B_m \text{ is not followed by a } Copy \text{ label and } i=m \\ \varepsilon & \text{if } B_1 \text{ is not preceded by a } Copy \text{ label and } i=0 \\ t & \text{if } B_{i+1} \text{ immediately precedes by } Copy(t) \text{ and } i<m \\ t & \text{if } B_m \text{ is followed by } Copy(t) \text{ and } i=m \end{cases}$$

The definition of the interpretation is given according to the following cases:

1. $s$ is a simple chain leading to a non-primitive node.
2. $s$ is a simple chain leading to an input channel.
3. $s$ is loop-ended chain of index $r$ leading to a non-primitive node.
4. $s$ is loop-ended chain of index $r$ leading to a multi-plicator.
5. $s$ is a simple chain leading to a generator.

In each of these cases we first define a sequence $At_1 \cdots At_m$ of automata and then we define $Int_s(A) = \sum_{i=1}^{i=m}\{At_i : B_i = A\}$

*Case* 1. $s = ap_1 A_1 q_1 \cdots p_n A_n$ is a simple chain leading to a non-primitive node. Define

$$At_i = \begin{cases} Test(\overline{q_i}, t_i);\ Decrement(\overline{p_i}, \overline{q_i}, i) & \text{if} \quad i<m \\ \textbf{output } d_{m-1} \textbf{ on port } \overline{p_m} & i=m \end{cases}$$

$$Int_s^1(A) = \sum_{i=1}^{i=m} \{At_i : B_i = A\}.$$

*Case* 2. $s = ap_1 A_1 q_1 \cdots p_n A_n q_n b$ is a simple chain leading to an input channel $b$. Define $At_i$ as $Test(\overline{q_i}, t_i)$; $Decrement(\overline{p_i}, \overline{q_i}, i)$, for $i = 1...m$.

$$Int_s^2(A) = \sum_{i=1}^{i=m} \{At_i : B_i = A\}.$$

*Case* 3. $s = ap_1 A_1 q_1 \cdots p_n A_n$ is a loop-ended chain of index $r$ leading to a non-primitive node.

Define

$$At_i = \begin{cases} \textbf{output } d_{m-1} \textbf{ on port } \overline{p_m};\ Test(\overline{q_r}, t_i);\ Decrement(\overline{p_r}, \overline{q_r}, r) & i \in \{r, m\} \\ Test(\overline{q_i}, t_i);\ Decrement(\overline{p_i}, \overline{q_i}, i) & \text{otherwise} \end{cases}$$

$$Int_{s,r}^3(A) = \sum_{i=1}^{i=m} \{At_i : B_i = A\}.$$

*Case* 4. $s = ap_1A_1q_1 \cdots p_nA_nq_n$ is a loop-ended chain of index $r$ leading to a multi-plicator.

Define $At_i$ as

$$\begin{cases} \textbf{output } d_m \textbf{ on port } \overline{p_r};\ Test(\overline{q_r},\ t_i);\ Decrement(\overline{p_r},\ \overline{q_r},\ r); \\ \qquad Decrement(\overline{p_r},\ \overline{q_r},\ r) & i \in \{r, m\} \\ Test(\overline{q_{r-1}},\ t_i);\ Test(\overline{q_{r-1}},\ d_m);\ Decrement(\overline{p_{r-1}},\ \overline{q_{r-1}},\ r-1) & i = r-1 \\ Test(\overline{q_i},\ t_i);\ Decrement(\overline{p_i},\ \overline{q_i},\ i) & \text{otherwise} \end{cases}$$

$$Int^4_{s,r}(A) = \sum_{i=1}^{i=m} \{At_i^{(k)} : B_i = A\}.$$

*Case* 5. $s$ is a simple chain leading to $Gen(t)$.

In this case we define a sequence of interpretations $\{Int_s^{(k)}\}_{k=1}^{\infty}$.

Let $t^{(k)}$ be the prefix of length $k$ of a stream $t$ ($t^{(k)} = t$, if $k$ is greater than the length of $t$).

Define

$$At_i^{(k)} = \begin{cases} Test(\overline{q_i},\ t_i);\ Decrement(\overline{p_i},\ \overline{q_i},\ i) & \text{if} \quad i < m \\ Test(\overline{q_m},\ t^{(k)});\ \textbf{output } d_{m\text{-}1} \textbf{ on port } \overline{p_m} & \text{if} \quad i = m \end{cases}$$

$$Int_i^{(k)}(A) = = \sum_{i=1}^{i=m} \{At_i^{(k)} : B_i = A\}.$$

In the following Lemma we use the notation $t :: s$ for the concatenation of streams $s$ and $t$.

LEMMA 50. *Let $N$ be a net in normal form. Assume that distinct values $d_0, d_1 \cdots$ do not appear in the parameters of copy and generator nodes of $N$.*

1. *if $s$ has the form $s = ap_1A_1q_1 \cdots p_nA_n$ and $A_n$ is a non-primitive label, then $N$ can produce $t_0 :: d_0$ on channel $a$ under the interpretation $Int^1_s$ iff $s$ is a simple chain in $N$.*

2. *if $s$ has the form $s = ap_1A_1q_1 \cdots p_nA_nq_nb$ and $s$ contains $m > 0$ non-primitive labels, then $N$ can produce $t_0 :: d_0$ on channel $a$ under the interpretation $Int^2_s$ when $d_m$ is received on channel $b$ iff $s$ is a simple chain of $N$.*

3. *if $s$ has the form $s = ap_1A_1q_1 \cdots p_nA_n$ and $A_n$ is a non-primitive label, then $N$ can produce $t_0 :: d_0$ on channel $a$ under the interpretation $Int^3_{s,r}$ iff $s$ is a loop-ended chain of index $r$ in $N$.*

4. *if $s$ has the form $s = ap_1A_1q_1 \cdots p_nA_nq_n$ and $A_n$ is a non-primitive label, then $N$ can produce $t_0 :: d_0$ on channel $a$ under the interpretation $Int^3_{s,r}$ iff $s$ is a loop-ended chain of index $r > 1$ in $N$.*

5. *if $s$ has the form $ap_1A_1q_1 \cdots p_nA_nq_n$ and $A_n$ is a non-primitive label, then $N$ can produce $t_0 :: d_m :: d_0$ on channel $a$ under the interpretation $Int^4_{s,0}$ iff $s$ is a loop-ended chain of index* 1 *in $N$.*

6. *if $s$ has the form $ap_1A_1q_1 \cdots p_nA_nq_n1Gen(t)$ and $t \neq \varepsilon$ and $n > 0$ then $N$ can produce $t_0 :: d_0$ on channel $a$ under the interpretation $Int^{(k)}_s$ iff there exists $t'$ such that $t^{(k)}$ is a prefix of $t'$ and either $ap_1A_1q_1 \cdots p_nA_nq_n1Gen(t')$ is a simple chain in $N$ or $ap_1A_1q_1 \cdots p_nA_nq_n1Copy(t')$ is a simple chain of $N$.*

*Proof.* We will prove only Lemma 50(1). Lemma 50(2–6) can be proved in a similar way.

(1) Let $B_1 \cdots B_m$ be the subsequence of non-primitive labels in $s$ and let $pos(i)$ be the position of $B_i$ in chain $s$. The following invariant can be shown by the induction on $k$.

*Invariant.* If $d_{m-k}$ is produced on a port $p$ of a node $v$ in a net $N$ under the interpretation $Int^1_s$, then $p = \overline{p_{m-k+1}}$, and $v$ is labeled by $B_{m-k+1}$ and $N$ has different nodes $v = v_{pos(m-k+1)}$, $v_{pos(m-k+1)+1} \cdots v_n$ labeled by $B_{m-k+1}$, $A_{pos(m-k+1)+1} \cdots A_m$ respectively and for every $j = pos(m-k)+1 \cdots n$ either there is a channel from port $p_j$ of $v_j$ to port $q_{j-1}$ of $v_{j-1}$ or there is a node $u_j$ labeled by a multi-plicator and its input port is connected to port $p_j$ of $v_j$ and its output port is connected to port $q_{j-1}$ of $v_{j-1}$.

Lemma 50(1) follows from this invariant. ∎

Now we are ready to prove

LEMMA 51. *Let $N_1$ and $N_2$ be nets in normal form. If $N_1$ and $N_2$ are equivalent under all interpretations then*

1. *$N_1$ and $N_2$ have the same set of loop-ended chains of index $r$.*
2. *$N_1$ and $N_2$ have the same set of simple proper chains.*
3. *$N_1$ and $N_2$ have the same set of simple proper strings.*

*Proof.* (1) A loop-ended chain has one of the following two forms: $ap_1A_1q_1 \cdots p_nA_nq_n$ or $ap_1A_1q_1 \cdots p_nA_n$, where $A_n$ is a non-primitive label. Since the nets have the same I–O behavior, Lemma 50(3–5) implies that they have the same set of loop-ended chains of index $r$.

(2) A simple proper chain has one of the following forms:

*Case* 1. $ab$

*Case* 2. $a1Copy(t)2b$

*Case* 3. $ap_1A_1q_1 \cdots p_nA_n$ and $A_n$ is a non-primitive label

*Case* 4. $ap_1A_1q_1 \cdots p_nA_nq_nb$ and contains $m > 0$ non-primitive labels

*Case* 5. $ap_1A_1q_1 \cdots p_nA_nq_n1Gen(t)$ and $n > 0$ and $t \neq \varepsilon$

*Case* 6. $ap_1A_1q_1 \cdots p_nA_nq_n1Gen(\varepsilon)$ and $n > 0$

*Case* 7. $a1Gen(t)$

Case 1 and Case 2. First note that by reduction $S_8$, the parameter $t$ in not equal $\varepsilon$ in case 2. Let $Int_{\varnothing}$ be an interpretation which assigns to all non-primitive labels automata which never produce any output.

It is clear that $ab$ (respectively $a1Copy(t)2b$) is a chain of $N_i$ iff under the interpretation $Int_{\varnothing}$ when $d_0$ is supplied to the channel $b$, net $N_i$ can produce $d_0$ (respectively $t :: d_0$) on channel $a$.

By the assumption of the Lemma, $N_1$ and $N_2$ have the same I–O behavior under all interpretations, hence: (1) $ab$ is a chain of $N_1$ iff it is a chain of $N_2$; (2) $a1Copy(t)2b$ is a chain of $N_1$ if it is a chain of $N_2$.

Case 3. This case follows from Lemma 50(1).

Case 4. This case follows from Lemma 50(2).

Case 5. $s$ leads to a generator and has form $s = ap_1A_1q_1 \cdots p_nA_nq_n1Gen(t)$.

We assume that $s$ is a simple chain of $N_1$ and will show that $s$ is a simple chain of $N_2$. Symmetrical arguments show that if $s$ is a simple chain of $N_2$ then it is a simple chain of $N_1$.

By the reduction rules $S_1$ and $S_5$, label $A_n$ should be non-primitive. Therefore, $s' = ap_1A_1q_1 \cdots p_nA_n$ is also a simple proper chain of $N_1$.

By case 3, chain $s'$ should belong to both nets. Note that $q_n$ is an input port of $A_n$ therefore $s'$ can be extended to a simple proper or a loop-ended chain $s''$ in $N_2$ which has one of the following forms:

(a) $s'' = ap_1A_1q_1 \cdots p_nA_nq_nb$ or $s'' = ap_1A_1q_1 \cdots p_nA_nq_n1Copy(t')2b$.

(b) $s'' = ap_1A_1q_1 \cdots p_nA_nq_np_{n+1}B$ or $s'' = ap_1A_1q_1 \cdots p_nA_nq_n1Copy(t)2p_{n+1}B$ and $B$ is a non-primitive label.

(c) $s'' = ap_1A_1q_1 \cdots p_nA_nq_n$ is a loop-ended chain of $N_2$

(d) $s'' = ap_1A_1q_1 \cdots p_nA_nq_n1Gen(t')$.

In sub-cases (a, b, c), $s''$ cannot be a chain of $N_1$. And these sub-cases will contradict our assumption that the nets have the same I–O behavior under all interpretations. Indeed, in sub-case (a) the nets will exhibit different I–O behaviors under $Int^2_{s''}$; in sub-case (b) they will exhibit different I–O behaviors under $Int^1_{s''}$ or one of the interpretations $Int^3_{s'', r}$ for loop-ended chains; in sub-case (c) the nets will exhibit different I–O behaviors under $Int^4_{s'', r}$.

Therefore, only sub case (d) is possible, and we are going to show that $t = t'$.

For every $k$, the nets have the same I–O behavior under the interpretations $Int^{(k)}_{s}$. Therefore, by Lemma 50(6), $t$ is a prefix of $t'$. For every $k$, the nets have the same I–O behavior under the interpretations $Int^{(k)}_{s'}$. Therefore, by Lemma 50(6), $t'$ is a prefix of $t$. Hence $t = t'$.

Therefore $s = s''$ and $s$ is a simple chain of $N_2$.

Cases 6 and 7 are treated similarly to case 5 and their proofs are left to the reader.

(3) follows immediately from Definition 6 and Lemma 51(2). ∎

Now for two strings we construct an interpretation which will distinguish the nets in which these strings meet from other nets. We start with a generic example and then provide a general construction. But first let us agree on the following

*Notations.*

- $d_0, d_1, \ldots$ is a sequence of distinct data values.
- *Decrement* $-$ *two*$(q, p, n)$ describes an automaton which first keeps reading from port $q$ and ignores all values until it finds $d$ in the above sequence. Then if $d = d_n$ it outputs $d_{n-2}$ on port $p$.

EXAMPLE 5. Let $s = a2A35D61C$ and $s' = b7B54A67C$ be two strings. Assume that $s$ and $s'$ are simple strings of a net $N$. Then the path which corresponds to $s$ in $N$ passes through distinct non-primitive nodes $v_1, v_2, v_3$. Similar, the path which corresponds to $s'$ in $N$ passes through distinct non-primitive nodes $v'_1, v'_2, v'_3$. By the definition, these strings meet iff nodes $v_3$ and $v'_3$ coincide. Nodes $v_1$ and $v'_2$ have the same label $A$, so they might coincide in $N$.

Define a sequence of automata $At_1$, $At_2$, $At_3$ and a sequence of automata $At'_1$, $At_2$, $At'_3$ as follows:

$At_3$ corresponds to $v_3$ and it outputs $d_4$ on port 1.

$At_2$ corresponds to $v_2$ and it behaves like *Decrement* $-$ *two*$(6, 5, 4)$.

$At_1$ corresponds to $v_1$ and it behaves like *Decrement* $-$ *two*$(3, 2, 2)$.

$At'_3$ corresponds to $v'_3$ and it outputs $d_5$ on port 7.

$At'_2$ corresponds to $v'_2$ and it behaves like *Decrement* $-$ *two*$(6, 4, 5)$.

$At'_1$ corresponds to $v_1$ and it behaves like *Decrement* $-$ *two*$(5, 7, 3)$.

Define the interpretation *Int* as follows:

$Int(A) = At_1 + At'_2 + At_1;At'_2$, because $v_1$ and $v'_2$ are labeled by $A$.

$Int(B) = At'_1$,because $v'_2$ is the only node labeled by $B$.

$Int(C) = At_3 + At'_3$, because $v_3$ and $v'_3$ are labeled by $C$ and they are the last nodes of strings $s$ and $s'$.

$Int(D) = At_2$, because $v_2$ is the only node labeled by $D$.

It is easy to see that the following holds.

*Observation* 5. Let $N$ be a net in normal form. Assume that a net $N$ has simple strings $s$ and $s'$ and that $d_0, \ldots, d_5$ do not appear in the parameters of the primitive nodes of $N$. Net $N$ can produce $d_0$ on channel $a$ and $d_1$ on channel $b$ under the above interpretation *Int* iff $s$ and $s'$ do not meet in $N$.

Let us now describe the general construction.

Let $s = ap_1 B_1 q_1 p_2 B_2 \dot{p}_m B_m$ and $s' = a'p'_1 B'_1 q'_1 p'_2 B'_2 \dot{p}'_{m'} B'_{m'}$ be two strings. Assume that $B_m$ and $B'_{m'}$ are non-primitive labels. We are going to define an interpretation $Int_{s,\, s'}$ which will distinguish between the nets in which $s$ and $s'$ meet and other nets.

First, define two sequences $\{At_1 \cdots At_m\}$ and $\{At'_1 \cdots At'_{m'}\}$of automata.

$$At_i = \begin{cases} \mathit{Decrement-two}(\overline{p_i}, \overline{q_i}, 2i) & \text{if} \quad i < m \\ \textbf{output } 2m\text{-}2 \textbf{ on port } p_m & \quad\; i = m \end{cases}$$

$$At'_j = \begin{cases} \mathit{Decrement-two}(\overline{p'_1}, \overline{q'_2}, 2i+1) & \text{if} \quad j < m' \\ \textbf{output } 2n\text{-}1 \textbf{ on port } p'_n & \quad\; i = m' \end{cases}$$

Second, define the interpretation $\mathit{Int}_{s,\, s'}(B)$ as

$$\sum \{At_i : B_i = B \text{ and } i \leqslant m\} + \sum \{At'_j : B'_j = B \text{ and } j \leqslant m'\}$$

$$+ \sum \{At_i;\ At'_j : B_i = B = B'_j \text{ and } i < m \text{ and } j < m'\}$$

LEMMA 52. *Let* $s = ap_1 B_1 q_1 p_2 B_2 \dot{p}_m B_m$ *and* $s' = a'p'_1 B'_1 q'_1 p'_2 B'_2 \dot{p}'_{m'} B'_{m'}$ *be two simple strings of a net N. Assume that* $B_m$ *and* $B'_{m'}$ *are non-primitive labels and that* $d_0, d_1 \cdots$ *do not appear in the parameters of copy and generator nodes of N. Under the interpretation* $\mathit{Int}_{s,\, s'}$ *net N can pass* $d_0$ *on the output channel a and* $d_1$ *on the output channel a' iff s and s' do not meet in N.*

*Proof.* Let $\mathit{Int}_s$ be the interpretation which corresponds to the first sum in the definition of $\mathit{Int}_{s,\, s'}$, i.e., $\mathit{Int}_s(B) = \sum \{At_i : B_i = B \text{ and } i \leqslant m\}$. Let $\mathit{Int}_{s'}$ be the interpretation which corresponds to the second sum in the definition of $\mathit{Int}_{s,\, s'}$. It is clear that $d_0$ (respectively $d_1$) can pass over channel $a$ (respectively channel $a'$) of a net $N$ under the interpretation $\mathit{Int}_s$ (respectively $\mathit{Int}_{s'}$) iff $s$ (respectively $s'$) is a simple string of $N$. Hence, if $s$ and $s'$ are simple strings of $N$ then there exists a run in which $d_0$ is passed over $a$ and there exists a run in which $d_1$ is passed over $a'$.

We have to show that $s$ and $s'$ do not meet in $N$ iff there exists a run in which $d_0$ passes over $a$ and $d_1$ passes over $a'$.

Let $u_i$ be the $i$th non-primitive node on the path corresponding to $s$ and $u'_j$ be the $j$th non-primitive node on the path corresponding to $s'$. Node $u_i$ is labeled by $B_i$ and node $u'_j$ is labeled by $B'_j$.

It is easy to show that for any run (a) $d_{2(i-1)}$ can pass over port $p_i$ of $u_i$ iff for every $k > i$, $d_{2(k-1)}$ have been passed over port $p_k$ of $u_k$, and (b) $d_{2(j-1)+1}$ can pass over port $p'_j$ of $u'_j$ iff for every $l > j$, $d_{2(l-1)+1}$ have been passed over port $p'_l$ of $u'_l$.

If the strings $s$ and $s'$ meet in $N$, then there exists a node $v$ such that $v = u_m = u'_{m'}$. Let $B$ be the label of the node $v$. It is clear that $B = B_m = B_{m'}$ and the interpretation assigns to this label the automaton $\sum \{At_i : B_i = B \text{ and } i \leqslant m\} + \sum \{At'_j : B'_j = B \text{ and } j \leqslant m'\} + \sum \{At_i; At'_j : B_i = B = B'_j \text{ and } i < m \textit{ and } j < m'\}$.

In no run this automaton can produce $d_{2(m-1)}$ over port $p_m$ and $d_{2(m'-1)+1}$ over port $p'_{m'}$. Indeed, if it produces $d_{2(m-1)}$ it should choose the behavior of $At_m$; if it produces $d_{2(m'-1)+1}$ it should choose the behavior of $At'_{m'}$. Therefore, in no run it is possible that $d_{2(m-1)}$ passes over port $p_m$ of $u_m$ and $d_{2(m'-1)+1}$ passes over port $p'_{m'}$ of $u'_{m'}$. Therefore, it is impossible that $d_0$ is passed on the channel $a$ and $d_1$ is passed on the channel $a'$.

It remains to show that if $s$ and $s'$ do not meet, then there is a run such that $d_0$ is passed on $a$ and $d_1$ is passed on $a'$.

We are going to describe for the nodes $u_i$, and $u'_j$ what behavior it should choose and how the work of the net should be scheduled.

If $u_i$ does not lie on the path for $s'$ it should behave like $At_i$. If $u'_j$ does not lie on the path of $s$ it should behave like $At'_j$. If a node $v$ lies on the path for $s$ and on the path for $s'$ then $v = u_i = u'_j$ for some $i$ and $j$. In this case $v$ should behave like $At_i; At'_j$. The following global schedule will insure that $d_0$ will pass over $a$ and $d_1$ will pass over $a'$. First, activate the automata on path $s$. Eventually $d_0$ will pass over $a$. At this moment the $j$th node on the path $s'$ can behave like $At'_j$ $(j = 1 \cdots m)$. We will activate these nodes and eventually, $d_1$ will pass over $a'$. ▮

Finally, Proposition 13(1–2) follows from Lemma 51. To prove Proposition 13(3) note that if proper strings $s$ and $s'$ meet in $N$ then one of the following cases holds:

1. $s$ and $s'$ lead to the same input channel $b$; in this case $s$ has form $ap_1B_1q_1p_2B_2 \cdots p_mB_mb$ and $s'$ has form $a'p'_1B'_1q'_1p'_2B'_2 \cdots p'_{m'}B'_{m'}b$.

2. $s$ and $s'$ lead to the node with non-primitive label; in this case $s = ap_1B_1q_1p_2B_2 \cdots p_mB_m$ and $s' = a'p'_1B'_1q'_1p'_2B'_2 \cdots p'_{m'}B'_{m'}$ and $B_m$, $B'_{m'}$ are non-primitive labels.

3. $s$ and $s'$ lead to a generator node; in this case $s = ap_1B_1q_1p_2B_2 \cdots p_mB_mq_m1Gen(t)$ and $s' = a'p'_1B'_1q'_1p'_2B'_2 \cdots p'_{m'}B'_{m'}q_{m'}1Gen(t)$, and $q_m = q'_{m'}$ and $B_m$, $B'_{m'}$ are non-primitive labels and simple proper chains $ap_1B_1q_1p_2B_2\dot{p}_mB_m$ and $a'p'_1B'_1q'_1p'_2B'_2\dot{p}'_{m'}B'_{m'}$ meet in $N$.

In case 1, Lemma 51(3) implies that that $s$ and $s'$ meet in $N_1$ iff they meet in $N_2$. In case 2 and case 3, Lemma 52 and Lemma 51(3) imply that $s$ and $s'$ meet in $N_1$ iff they meet in $N_2$. Therefore, Proposition 13 holds.

## ACKNOWLEDGMENTS

I am grateful to B. A Trakhtenbrot for many helpful suggestions on the contents and the presentation of the paper. I would like also to thank the anonymous referees for their constructive proposals.

Received December 14, 1995; final manuscript received May 19, 1997

## REFERENCES

1. Abramsky, S. (1990), A generalized Kahn principle for abstract asynchronous networks, *in* "Mathematical Foundations of Programming Languages Semantics," Lecture Notes in Computer Science, Vol. 442, Springer-Verlag, Berlin/New York.
2. Brock, J. D., and Ackerman, W. B. (1987), Scenarios: A model of non-determinate computation, *in* "Formulization of Programming Concepts," Lecture Notes in Computer Science, Vol. 107, pp. 252–259, Springer-Verlag, Berlin/New York.
3. Cormen, T. H., Leiserson, C. E., and Rivest, R. L. (1991), "Introduction to Algorithms," MIT Press, Cambridge, MA.

4. Courcelle, B., Kahn, G., and Vuillemin, J. (1974), Algorithmes d'équivalence et de réduction a des expressions minimales dans une classe d'équations recursive simples, *in* "Proceedings of 2nd International Conference on Automata, Languages and Programming," Lecture Notes in Computer Science, Vol. 14, Springer-Verlag, Berlin/New York.

5. Dennis, J. B., and Foseen, J. B. (1973), "Introduction to Dataflow Schemas," MIT, Computation Structure Group Memo 81-1.

6. Faustini, A. (1982), An operational semantics for pure dataflow, *in* Lecture Notes in Computer Science, Vol. 140, pp. 212–224, Springer-Verlag, Berlin/New York.

7. Girard, J-Y. (1987), "Proof Theory and Logical Complexity," Bibliopolis, Naples.

8. Greibach, S. A. (1975), "Theory of Program Structures: Schemes, Semantics, Verification," Lecture Notes in Computer Science, Vol. 36, Springer-Verlag, Berlin/New York.

9. Gaifman, H., and Pratt, V. (1987), Partial order models of concurrency and the computation of functions, *in* "Proceedings, Symposium on Logic in Computer Science," pp. 72–85, IEEE Comput. Soc., Los Alamitos, CA.

10. Jonsson, B. (1989), A fully abstract model for dataflow networks, *in* "Proceedings, 16th ACM Symposium on Principles of Programming Languages," pp. 155–165.

11. Kahn, G. (1974), The semantics of a simple language for parallel programming, *in* "Information Processing 74," pp. 471–475, North-Holland, Amsterdam.

12. Kok, J. N. (1987), A fully abstract semantics for data flow nets, *in* Lecture Notes in Computer Science, Vol. 259, pp. 351–368, Springer-Verlag, Berlin/New York.

13. Lynch, N., and Tuttle, M. (1987), Hierarchical correctness proof for distributed algorithm, *in* "Proceedings of 6th ACM Symposium on Principles of Distributed Computing," pp. 137–151.

14. Lynch, N. A., and Stark, E. W. (1989), A proof of the Kahn principle for input/output automata, *Inform. and Comput.* **82**, 81–92.

15. Parrow, J. (1989), Structural and behavioral equivalences of networks, *in* "Proceedings of 16th International Conference on Automata, Languages and Programming," Lecture Notes in Computer Science, Vol. 443, Springer-Verlag, Berlin/New York, Full version, *Inform. and Comput.* **107**, 58–90 (1993).

16. Rabinovich, A. (1987), Pomset semantics is consistent with data flow semantics, *EATCS Bull.*, 107–117.

17. Rabinovich, A. (1992), Logic of trace languages, *in* "Proceedings of Third International Conference on Concurrency Theory," Lecture Notes in Computer Science, Vol. 630, Springer-Verlag, Berlin/New York.

18. Rabinovich, A. (1996), On schematological equivalence of dataflow networks, *Inform. and Comput.* **124**, 154–167.

19. Rabinovich, A., and Trakhtenbrot, B. A. (1988), Nets of processes and data flow, *in* "Proceedings of Rex Workshop on Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency," Lecture Notes in Computer Science, Vol. 354, pp. 574–602, Springer-Verlag, Berlin/New York.

20. Stark, E. (1992), A calculus of dataflow networks, *in* "Proceedings of 7th Symposium on Logic in Computer Science," pp. 125–136.