BEHAVIOUR OF ELEMENTARY NET SYSTEMS

A. Ehrenfeucht*

and

G. Rozenberg**

CU-CS-339-86 August, 1986

All correspondence to second author.

This research was supported by National Science Foundation grant number MCS 79-03838.

* University of Colorado, Department of Computer Science, Boulder, Colorado, USA.

** Institute of Applied Mathematics and Computer Science, University of Leiden, Leiden, The Netherlands.

Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author and do not necessarily reflect the views of the National Science Foundation.

BEHAVIOUR OF ELEMENTARY NET SYSTEMS

by

G. ROZENBERG

Department of Computer Science
University of Leiden
Leiden, The Netherlands

and

Department of Computer Science
University of Colorado at Boulder
Boulder, Colorado, U.S.A.

CONTENTS:

0. INTRODUCTION

The aim of this lecture is to discuss the behaviour of elementary net systems which were presented as a simple model of distributed systems in the lecture by P.S. Thiagarajan.

There are at least two reasons for formalizing the notion of the behaviour of an EN system. First of all one should be able to express formally what a given EN system does (how does it behave). Secondly, one gets a formal tool for comparing EN systems with each other.

Clearly, the question "What is the behaviour of an EN system ?" is strongly related to the question "How can the behaviour of an EN system be observed ?". The relationship of these two questions is strongly reflected in our lecture.

Actually our point of departure is to distinguish between *sequential* and *non-sequential* observations and then to identify records of such observations with the system behaviour.

In the sequential point of view each record of the behaviour of an EN system is a string of event occurrences (called a *firing sequence*) as registered by a sequential observer.

In the non-sequential point of view one can either record all non-sequential observations of event occurrences (called *firing traces*) or proceeding in a more detailed fashion one can record all non-sequential observations of event occurrences together with the resulting holdings of conditions; such records are

called *processes.*

During the lecture we discuss the two approaches and compare them with each other.

Our way of defining the behaviour of EN systems is not the only one discussed in the literature. Some of the other approaches will be discussed in the lectures by C. Fernandez, M. Jantzen, and R. Valk. Even within the above sketched methodology *we consider finite behaviours only ; consequently we will consider finite words and finite nets only.* Some aspects of extending our considerations to the infinite case (infinite stretches of behaviour) will be discussed in the lectures by C. Fernandez and R. Valk.

We end this introduction with the following remarks.

(1) Obviously, this lecture directly continues the lecture by P.S. Thiagarajan - hence we build on notions and notations introduced there.

(2) These notes are meant only as the underlying material for the lecture - considered on its own they may be incomplete.

1. PRELIMINARIES

Although we assume the reader to be familiar with basic mathematical notions (sets, graphs, bipartite graphs, node- and edge-labeled graphs, graph isomorphisms, etc.) we recall a number of them in this section in order to establish our notation and terminology and also because some of these notions are used in our paper in a specific way.

When dealing with sets *we will often identify a singleton set* $\{x\}$ *with its element* x - this however should not lead to confusion. For a set X, $|X|$ denotes its cardinality.

For an alphabet Σ, Σ^* denotes the set of all finite words over Σ and Λ denotes the *empty word.*

We will deal with various kinds of graphs. Instead of adopting one general definition, it will be more convenient to develop various notions of graphs and introduce a specific notation for each notion.

A (*directed*) *graph* is an ordered pair $g = (V, Y)$ where V is the set of *nodes* and $Y \subseteq V \times V$ is the set of *edges. Unless explicitly stated otherwise we deal with nonempty graphs only* (i.e., we assume that $V \neq \emptyset$). A (*directed*) *path* in g is a sequence of nodes $\pi = v_0, \cdots, v_n$, $n \geq 1$, such that $(v_i, v_{i+1}) \in Y$ for all $0 \leq i < n$; if $v_0 = v_n$ then π is a (*directed*) *circuit.* g is said to be *acyclic* iff no path in g is a circuit.

We discuss now other kinds of graphs used in these notes.

A *bipartite graph* is a triplet (V_1, V_2, Y) where $V_1 \neq \emptyset$, $V_2 \neq \emptyset$, $V_1 \cap V_2 = \emptyset$, $(V_1 \cup V_2, Y)$ is a graph and $Y \subseteq (V_1 \times V_2) \cup (V_2 \times V_1)$.

Remark 1.1. Our definition of a bipartite graph differs somewhat from the conventional definition mostly used in the literature : we consider the two sets partitioning the set of all nodes to be an ordered pair (V_1, V_2) so that we can distinguish between (and unambiguously refer to) them. This slight definitional difference is quite crucial when we use bipartite graphs to deal with nets. □

The terminology concerning paths is carried over to a bipartite graph (V_1, V_2, Y) by viewing it as a graph $(V_1 \cup V_2, Y)$.

Let $g = (V_1, V_2, Y)$ be a bipartite graph and let $W \in \{V_1, V_2\}$. The *W-contraction of* g, denoted $ctr_W(g)$, is the graph $\{(V_1 \cup V_2) - W, Y_W\}$, where $(v, v') \in Y_W$ iff there exists a $w \in W$ such that $(v, w) \in Y$ and $(w, v') \in Y$.

Let Σ be an alphabet. A *node* Σ*-labeled graph* is a triplet (V, Y, ϕ), where (V, Y) is a graph and $\phi : V \longrightarrow \Sigma$ is the (*node*) *labeling function.* A *node* Σ-labeled bipartite graph is a 4-tuple (V_1, V_2, Y, ϕ) where $(V_1 \cup V_2, Y, \phi)$ is a node Σ-labeled graph such that $\phi(V_1) \cap \phi(V_2) = \emptyset$. An *edge* Σ*-labeled initialized graph* is a triplet $g = (V, Y, v_{in})$ where $V \neq \emptyset$ is the set of *nodes,* $Y \subseteq V \times \Sigma \times V$ is the set of Σ*-labeled edges* and $v_{in} \in V$ is the *initial node; rlab*(g) denotes the set $\{\sigma \in \Sigma \mid (v, \sigma, v') \in Y \text{ for some } v, v' \in V\}$. In all the terminology above we may replace " Σ-labeled" by "labeled" whenever Σ is clear

from the context.

Let $g_1 = (V_1, Y_1, \phi_1)$ be a node Σ_1-labeled graph and $g_2 = (V_2, Y_2, \phi_2)$ a node Σ_2-labeled graph.

We say that g_1 and g_2 are *isomorphic,* written g_1 *isom* g_2 iff

there exists a bijection $\alpha : V_1 \longrightarrow V_2$ such that

$(\forall v \in V_1)[\, \phi_2(\alpha(v)) = \phi_1(v)\,]$ and

$(\forall v, v' \in V_1)[\, (v, v') \in Y_1$ iff $(\alpha(v), \alpha(v')) \in Y_2\,]$.

We say that g_1 and g_2 are *label-isomorphic* and write g_1 *lisom* g_2 iff there exist bijections $\alpha : V_1 \longrightarrow V_2$ and $\beta : \phi_1(V_1) \longrightarrow \phi_2(V_2)$ such that

$(\forall v \in V_1)[\, \phi_2(\alpha(v)) = \beta(\phi_1(v))\,]$ and

$(\forall v, v' \in V_1)[\, (v, v') \in Y_1$ iff $(\alpha(v), \alpha(v')) \in Y_2\,]$.

Let $g_1 = (V_1, V_2, Y_1, \phi_1)$ be a node Σ_1-labeled bipartite graph and $g_2 = (W_1, W_2, Y_2, \phi_2)$ a node Σ_2-labeled bipartite graph. We say that g_1 and g_2 are *label-isomorphic,* written g_1 *lisom* g_2 , iff there exist bijections

$\alpha : V_1 \cup V_2 \longrightarrow W_1 \cup W_2$ and $\beta : \phi_1(V_1 \cup V_2) \longrightarrow \phi_2(W_1 \cup W_2)$ such that

$(\forall v \in V_1 \cup V_2)[\, v \in V_1$ iff $\alpha(v) \in W_1\,]$,

$(\forall v \in V_1 \cup V_2)[\, \phi_2(\alpha(v)) = \beta(\phi_1(v))\,]$ and

$(\forall v, v' \in V_1 \cup V_2)[\, (v, v') \in Y_1$ iff $(\alpha(v), \alpha(v')) \in Y_2\,]$.

Let $g_1 = (V_1, Y_1, v_1)$ be an edge Σ_1-labeled initialized graph and $g_2 = (V_2, Y_2, v_2)$ be an edge Σ_2-labeled initialized graph. We say that g_1 and g_2 are *label-isomorphic,* written g_1 *lisom* g_2, iff there exist bijections

$\alpha : V_1 \longrightarrow V_2$ and $\beta : rlab(g_1) \longrightarrow rlab(g_2)$ such that

$\alpha(v_1) = v_2$ and

$(\forall v, v' \in V_1)(\forall \sigma \in rlab(g_1))[\ (v, \sigma, v') \in Y_1$ iff $(\alpha(v), \beta(\sigma), \alpha(v')) \in Y_2\ ]$.

In these notes we will deal with *partially ordered sets* (*posets*) and with *labeled partially ordered sets* (*labeled posets*).

A *poset* is a graph $g = (V, Y)$ such that Y is a transitive, antisymmetric and reflexive relation. A *Σ-labeled poset* is a node Σ-labeled graph $g = (V, Y, \phi)$ such that (V, Y) is a poset. Whenever Σ is clear from the context we may use the phrase "labeled" rather than "Σ-labeled".

With each acyclic graph $g = (V, Y)$ we associate the poset (V, Y^*) where Y^* is the transitive and reflexive closure of Y; this poset is denoted by $\leq_g$. Similarly with each Σ-labeled acyclic graph $g = (V, Y, \phi)$ we associate the labeled poset (V, Y^*, ϕ) which we also denote by $\leq_g$.

Remark 1.2. For the sake of notational simplicity we will sometimes write $x \leq_g y$ (where $x, y \in V$ and $V, g, \leq_g$ are as above) rather than to write $x\ Y^*\ y$; this however should not lead to confusion. □

In this lecture we will often consider the set of all (node-labeled) graphs isomorphic to a given (node-labeled) graph g ; this set is denoted by $\bar{g}$ and it is referred to as an *abstract* (*node−labeled*) *graph*. To avoid set theoretical difficulties we will assume that the nodes of all (node-labeled) graphs we consider are taken from one fixed countable set of elements.

2. ON THE SIMILARITY OF EN SYSTEMS

In this section we will be concerned with establishing criteria for the "similarity" of EN systems *without* defining first their behaviour. Rather, we do it by considering two basic components of an EN system - the *static structure* and the *distributed state space*.

For an EN system N its static structure is given by the underlying set N_N and so the notion of the structural similarity of EN systems is based on the standard notion of isomorphism of bipartite graphs.

Definition 2.1. Let $N_1 = (B_1, E_1, F_1, C_1)$ and $N_2 = (B_2, E_2, F_2, C_2)$ be EN systems. We say that N_1 and N_2 are *structurally similar,* denoted $N_1 \equiv N_2$, iff there exists a bijection $\alpha : X_{N_1} \rightarrow X_{N_2}$ such that

$(\forall x, y \in X_{N_1})[\ (\ x \in B_1$ iff $\alpha(x) \in B_2\ )$ and $((x,y) \in F_1$ iff $(\alpha(x), \alpha(y)) \in F_2\ )\ ]$

and

$\alpha(C_1) = C_2$. □

The distributed state space of an EN system N is given by its case graph. The notion of the case graph of an EN system was discussed already in the lecture by P.S. Thiagarajan - we given now the formal definition.

Definition 2.2. Let $N = (B, E, F, C_{in})$ be an EN system. The *case graph of* N, denoted CG_N, is the initialized edge-labeled graph (V, Y, v_{in}), where $V = C_N$, $Y = \{(C_1, U, C_2) \mid C_1[U > C_2\}$ and $v_{in} = C_{in}$. □

The notion of the state space similarity is based on the (label-) isomorphism of case graphs.

Definition 2.3. EN systems N_1, N_2 are *state space similar,* denoted

$N_1 \cong N_2$, iff CG_{N_1} *lisom* CG_{N_2}. □

Now we return to the "complement construction" discussed in the lecture by P.S. Thiagarajan. This construction converts an arbitrary EN system into a contact-free EN system and it was argued that the resulting system is "equivalent" to the original system. Now we can formalize this claim.

Theorem 2.1.

(1) Let N be an EN system and let N' be the EN system obtained by applying the complement construction to N. Then $N \cong N'$.

(2) For every EN system N there exists a contact-free EN system N' such that $N \cong N'$. □

In the sequel of these notes, unless explicitly stated otherwise, we will consider contact-free EN systems only. This restriction is quite essential and we will discuss it in more detail in Section 6.

3. SEQUENTIAL OBSERVATIONS OF EVENT OCCURRENCES

Perhaps the simplest way to define the behaviour of an EN system is through *sequential observers.* Each record of an observation by a sequential observer is a string of event occurrences (called a firing sequence). The collection of all such records represents the behaviour of the system.

Definition 3.1. Let $N = (B, E, F, C_{in})$ be an EN system. $\rho \in E^*$ is a *firing sequence of* N iff

either $\rho = \Lambda$

or $\rho = e_1 \cdots e_n$, $n \geq 1$, $e_1, \cdots, e_n \in E$ and there exists a sequence of cases $C_0, C_1, \cdots, C_n$ in C_N such that $C_0 = C_{in}$ and $C_{j-1}[e_j > C_j$ for $1 \leq j < n$. □

We will use $\mathbf{FS}(N)$ to denote the set of firing sequences of N .

Firing sequences can be interpreted as walks in the restricted state space of an EN system. In order to formalize this statement we need the following notion.

Definition 3.2. Let $N = (B, E, F, C_{in})$ be an EN system. The *sequential case graph of* N, denoted SCG_N , is the edge-labeled initialized graph (V, Y, v_{in}) where

$V = C_N$,

$Y = \{(C_1, U, C_2) \mid C_1[U > C_2$ and $|U| = 1\}$ and $v_{in} = C_{in}$. □

Now we can interpret firing sequences as follows:

ρ is a firing sequence of an EN system N iff ρ is the sequence of edge-labels corresponding to a path in SCG_N starting from its initial node (recall our convention that we often identify singleton sets with their elements).

Example 3.1. Consider the following EN system $N_1 = (B_1, E_1, F_1, C_1)$:

Figure 3.1

Then CG_{N_1} is as follows :

Figure 3.2

and SCG_{N_1} is as follows :

Figure 3.3

Since we have

$\{b_1, b_2\}[e_1 > \{b_3, b_2\}$,

$\{b_3, b_2\}[e_3 > \{b_3, b_4\}$,

$\{b_3, b_4\}[e_5 > \{b_3, b_2\}$, and

$\{b_3, b_2\}[e_4 > \{b_1, b_2\}$,

each of the following strings :

Λ , e_1 , e_1e_3 , $e_1e_3e_5$, $e_1e_3e_5e_4$

is a firing sequence of N_1 ; it is easily seen that each of these strings is the sequence of edge labels corresponding to a path in SCG_{N_1} . □

SCG_N has a finite number of nodes and a finite number of edges labeled by events from N (actually by singleton sets consisting of events from N - but this difference is neglected in our considerations). Hence one can consider SCG_N to be the transition diagram of a finite state automaton where the initial node corresponds to the initial state of the automaton and all the nodes (states) are terminal.

Thus the following result is obvious.

Theorem 3.1. For an EN system N, the set $\mathbf{FS}(N)$ is a prefix closed regular language. □

Clearly, representing the behaviour of an EN system N via its firing sequences provides an important insight - e.g., it tells us about all the "sequential walks" (trajectories) in the state space of N. However there are some serious problems concerning this representation. A firing sequence records an observation by a *sequential observer* of a system that (in general) is basically *non-sequential.* It may happen that within a firing sequence there is a subsequence $e_1 e_2$ while in the system considered (observed) these two occurrences are cconcurrent. Thus the linear order of event occurrences in a firing sequence does not necessarily reflect the causal order enforced by the system on event occurrences.

Consequently, the description of the system behaviour via firing sequences will be really valuable if the causal (in general partial) order of events can be

extracted from firing sequences. In order to do so one has to augment the "observational" description of the system with some structural information about the system (i.e. information based on the underlying net of the system only).

An elegant approach to this problem is provided by the *theory of traces.*

4. TRACES AND DEPENDENCY GRAPHS

In this section we will briefly review the basic notions of the theory of traces to the extent that they are needed for our purpose. More advanced issues of this theory will be discussed in the lecture by A. Mazurkiewicz.

Definition 4.1.

(i) Let Σ be an alphabet. An *independence relation* (*over* Σ) is a symmetric and irreflexive binary relation over Σ . A *dependence relation* (*over* Σ) is a symmetric and reflexive binary relation over Σ .

(ii) A *reliance alphabet* is a triple (Σ, I, D), where Σ is an alphabet, I is an independence relation over Σ , and D is a dependence relation over Σ such that $I \cup D = \Sigma \times \Sigma$ and $I \cap D = \emptyset$. □

An independence relation over Σ induces two important relations over Σ^* .

Definition 4.2. Let $Z = (\Sigma, I, D)$ be a reliance alphabet.

(i) The relation $\doteq_I \subseteq \Sigma^* \times \Sigma^*$ is defined by :

$(\forall \rho, \rho' \in \Sigma^*)[\, \rho \doteq_I \rho' \text{ iff}$

$(\exists \rho_1, \rho_2 \in \Sigma^*)(\exists (a, b) \in I)[\, \rho = \rho_1 a b \rho_2 \text{ and } \rho' = \rho_1 b a \rho_2 \,]\,]$.

(ii) The relation $\overset{*}{=}_I \subseteq \Sigma^* \times \Sigma^*$ is the least equivalence relation over Σ^* containing $\doteq_I$. □

In the notation as above we say that ρ and ρ' are *I-equivalent* iff $\rho \overset{*}{=}_I \rho'$. We use $[\rho]_I$ to denote the equivalence class of $\overset{*}{=}_I$ containing ρ and we use $\Theta(I)$

to denote the set of all equivalence classes of I.

Remark 4.1. In the above we have used the notations like $\doteq_I$, $\overset{*}{=}_I$, and $[\rho]_I$ which formally speaking are not correct; rather one should use the notations like $\doteq_Z$, $\overset{*}{=}_Z$, and $[\rho]_Z$, respectively. However we have decided on this slight abuse of notation for didactical reasons. In the context of a given reliance alphabet $Z = (\Sigma, I, D)$ some of the notions we will consider will explicitly depend on I while other will explicitly depend on D - to make this apparent we will subscript our notation for the former by I and for the latter by D . □

Example 4.1. Consider the reliance alphabet $Z_1 = (\Sigma_1, I_1, D_1)$, where $\Sigma_1 = \{a, b, c, d\}$ and

$I_1 = \{(a,b), (b,a), (a,d), (d,a), (b,c), (c,b)\}$,

and the string $\rho_1 = a\,b\,c\,a\,d$.

Then

$\rho_1 \doteq_{I_1} b\,a\,c\,a\,d \doteq_{I_1} b\,a\,c\,d\,a$ and consequently $\rho_1 \overset{*}{=}_{I_1} b\,a\,c\,d\,a$.

As a matter of fact

$[\rho_1]_{I_1} = \{abcad, bacad, bacda, abcda, acbda, acabd, acbad\}$. □

We are ready now to define the notions of a trace and a trace language.

Definition 4.3. Let $Z = (\Sigma, I, D)$ be a reliance alphabet. An element of $\Theta(I)$ is called a *trace* (*over* I) and a subset of $\Theta(I)$ is called a *trace language* (over I) . □

For a reliance alphabet $Z = (\Sigma, I, D)$, a string language $K \subseteq \Sigma^*$ may be partitioned by $\overset{*}{=}_I$ in either a "consistent" or an "inconsistent" way.

Definition 4.4. Let $Z = (\Sigma, I, D)$ be a reliance alphabet and let $K \subseteq \Sigma^*$. We say that K is I - *consistent* iff

$(\forall t \in \Theta(I))[$ either $t \cap K = \emptyset$ or $t \subseteq K$] . □

Example 4.2. Consider the reliance alphabet Z_1 from Example 4.1 and consider the language $K = \{abcad, acabd, daa\}$. Then K is *not* I_1-consistent because $[\rho_1]_{I_1} \cap K \neq \emptyset$ while $[\rho_1]_{I_1} \not\subseteq K$ (where $\rho_1 = abcad$).

On the other hand,

$K' = \{abcad, bacad, bacda, abcda, acbda, acabd, acbad, cbd, bcd\}$

is Z_1-consistent since $K' = [\rho_1]_{I_1} \cup [cbd]_{I_1}$. □

Given a reliance alphabet $Z = (\Sigma, I, D)$ a string language $K \subseteq \Sigma^*$ induces a trace language over Z as follows.

Definition 4.5. Let $Z = (\Sigma, I, D)$ be a reliance alphabet and let $K \subseteq \Sigma^*$. The *trace language of* K (*over* I) denoted $[K]_I$ is the trace language

$\{[x]_I \mid x \in K\}$. □

Now we can use classes of string languages to define classes of trace languages. In particular we can transfer the notion of regularity of string languages into the framework of trace languages as follows.

Definition 4.6. Let $Z = (\Sigma, I, D)$ be a reliance alphabet and let $T \subseteq \Theta(I)$. □

We say that T is *regular* iff there exists a regular $K \subseteq \Sigma^*$ such that $T = [K]_I$.

Example 4.3. Let Z_1, K, K' and ρ_1 be as in Example 4.2. Then

$[K]_I = \{[\rho_1]_I, [daa]_I\}$ and $[K']_I = \{[\rho_1]_I, [cbd]_I\}$;

since both K and K′ are regular (even finite),

$[K]_I$ and $[K']_I$ are regular trace languages. □

Each trace may be viewed as an abstract acyclic node-labeled directed graph and consequently as an abstract labeled poset. This fact is crucial in applying the theory of traces to define the non-sequential behaviour of EN systems.

Definition 4.7. Let $Z = (\Sigma, I, D)$ be a reliance alphabet and let $\rho \in \Sigma^*$.

(i) If $\rho = \Lambda$ then the *canonical dependency graph of* ρ is the empty graph and the *canonical labeled poset of* ρ is the empty poset.

(ii) Let $\rho = \sigma_1 \ldots \sigma_n$ for some $n \geq 1, \sigma_1, \ldots, \sigma_n \in \Sigma$.

The *canonical dependency graph of* ρ is the Σ-labeled graph (V, Y, ϕ) defined by:

(1) $V = \{1, \ldots, n\}$,

(2) $(\forall i \in \{1, \ldots, n\})[\ \phi(i) = \sigma_i\ ]$,

(3) $(\forall i,j \in \{1,\ldots,n\})[(i,j) \in Y$ iff $(i < j$ and $(\sigma_i, \sigma_j) \in D)]$.

The *canonical labeled poset of* ρ is the labeled poset determined by the canonical dependency graph of ρ. □

The canonical dependency graph of ρ is denoted by $< \rho >_D$ and the canonical labeled poset of ρ is denoted by $<< \rho >>_D$.

Remark 4.2. It is important to notice that while D is reflexive, $< \rho >_D$ is irreflexive. □

The basic property of the above construction is the following.

Lemma 4.1. Let $Z = (\Sigma, I, D)$ be a reliance alphabet and let $\rho, \rho' \in \Sigma^*$. Then

$< \rho >_D$ *isom* $< \rho' >_D$ iff $[\rho]_I = [\rho']_I$. □

Thus for a trace $t = [\rho]_I \in \Theta(I)$, $< \rho >_D$ is an "isomorphic invariant" of t - it does not depend (up to an isomorphism) on the choice of a representative of t.

Example 4.4. Let Z_1 and ρ_1 be as in Example 4.1.

Then $< \rho_1 >_{D_1}$ is as follows :

Figure 4.1

For $\rho_2 = \text{bacda}$, $< \rho_2 >_{D_1}$ is as follows :

Figure 4.2

We notice that $\rho_1 \overset{*}{=}_I \rho_2$ (see Example 4.1) and indeed, according to Lemma 4.1, we have $< \rho_1 >_{D_1}$ *isom* $< \rho_2 >_{D_1}$; to see this consider the bijection α between the nodes of $< \rho_1 >_{D_1}$ and the nodes of $< \rho_2 >_{D_1}$ defined by: $\alpha(1) = 2, \alpha(2) = 1, \alpha(3) = 3, \alpha(4) = 5$ and $\alpha(5) = 4$.

Clearly, $<< \rho_1 >>_{D_1}$ is as follows :

Figure 4.3

while $<< \rho_2 >>_{D_1}$ is as follows :

Figure 4.4

□

Definition 4.8. Let $Z = (\Sigma, I, D)$ be a reliance alphabet

(i) Let $t \in \Theta(I)$.

The *abstract dependency graph of* t, denoted $adg(t)$, is the abstract node-labeled graph $\overline{< \rho >_D}$, where $\rho \in t$.

The *abstract labeled poset of* t , denoted $alp(t)$, is the abstract labeled poset $\overline{<< \rho >>_D}$, where $\rho \in t$.

(ii) Let $T \subseteq \Theta(I)$.

The *set of abstract dependency graphs of* T, denoted **ADG**(T), is the set $\{adg(t) \mid t \in T\}$.

The *set of abstract labeled posets of* T, denoted **ALP**(T), is the set $\{alp(t) \mid t \in T\}$. □

Remark 4.3.

(1) Note that Lemma 4.1 guarantees that the notions of the abstract dependency graph and of the abstract labeled poset of a trace are well-defined; i.e., given $\rho, \rho' \in t$, $\overline{< \rho >_D} = \overline{< \rho' >_D}$ and $\overline{<< \rho >>_D} = \overline{<< \rho' >>_D}$.

(2) Since we are mostly interested in *abstract* dependency graphs and *abstract* labeled posets we will often skip the adjective "abstract" and talk simply about dependency graphs and labeled posets of traces. Whenever we need to talk about a representative of the abstract dependency graph (abstract labeled poset) of a trace we will use the adjective "concrete" ; thus a concrete dependency graph of a trace t is an element of $adg(t)$ - in particular, $< \rho >_D$ for a ρ in t is a concrete dependency graph of t . □

5. EXTRACTING CAUSAL ORDERS FROM SEQUENTIAL OBSERVATIONS

We return now to the problem of extracting causal order of events from firing sequences using the theory of traces. The obvious first step in this direction is to associate an independence relation with an EN system.

Definition 5.1. Let $N = (B, E, F, C_{in})$ be an EN system. The *independence relation induced by* N, denoted I_N , is the binary relation over E defined by:

$(\forall e_1, e_2 \in E)[(e_1, e_2) \in I_N$ iff $({}^\bullet e_1 \cup e_1^\bullet) \cap ({}^\bullet e_2 \cup e_2^\bullet) = \emptyset]$.

The *reliance alphabet induced by* N, denoted Z_N, is the triplet (E, I_N, D_N) where $D_N = E \times E - I_N$ is the *dependence relation induced by* N . □

One can easily verify that Z_N is indeed a reliance alphabet (i.e. that I_N is a symmetric and irreflexive relation) .

The independence relation I_N for an EN system N is well-chosen in the following sense.

Theorem 5.1. Let N be an EN system. Then $\mathbf{FS}(N)$ is I_N-consistent. □

Hence if a firing sequence ρ is observable in N then so is every firing sequence that is I_N-equivalent with ρ !

Example 5.1. Consider the EN system N_1 from Example 3.1. Then

$I_{N_1} = \{(e_1,e_3), (e_3,e_1), (e_1,e_5), (e_5,e_1), (e_3,e_4), (e_4,e_3), (e_4,e_5), (e_5,e_4)\}$,

$D_{N_1} = (E_1 \times E_1) - I_{N_1}$,

and

$Z_{N_1} = (E_1, I_{N_1}, D_{N_1})$.

Clearly, $e_1 e_3 e_5 e_4 e_2 \in \mathbf{FS}(N_1)$.

It is easily verified that

$[e_1e_3e_5e_4e_2]_{I_{N_1}} = \{e_3e_5e_1e_4e_2, e_3e_1e_5e_4e_2, e_1e_3e_5e_4e_2, e_3e_1e_4e_5e_2, e_1e_3e_4e_5e_2,$

$e_1e_4e_3e_5e_2\}$.

By Theorem 5.1, $[e_1e_3e_5e_4e_2]_{I_{N_1}} \subseteq \mathbf{FS}(N_1)$ and indeed it is easily verified that each element of $[e_1e_3e_5e_4e_2]_{I_{N_1}}$ is the sequence of edge labels corresponding to a path in SCG_{N_1} (see Example 3.1) and thus is a firing sequence of N_1 . □

Now we can transform the set of firing sequences of an EN system into its set of traces, the set of abstract dependency graphs and the set of abstract labeled posets.

Definition 5.2. Let N be an EN system.

(i) The set of *firing traces of* N, denoted $\mathbf{FT}(N)$, is the trace language $[\mathbf{FS}(N)]_{I_N}$.

(ii) The set of *abstract firing dependency graphs of* N, denoted $\mathbf{AFD}(N)$, is the set $\mathbf{ADG}(\mathbf{FT}(N))$.

(iii) The set of *abstract firing labeled posets of* N, denoted $\mathbf{AFLP}(N)$, is the set $\mathbf{ALP}(\mathbf{FT}(N))$.

Example 5.2. Let N_1 be the EN system from Example 3.1. The following are some of the concrete firing dependency graphs of N_1 (see Example 5.1 for D_{N_1}) :

Figure 5.1

The corresponding concrete labeled posets are:

Figure 5.2

The following result follows directly from Theorem 3.1.

Theorem 5.2. $\mathbf{FT}(N)$ is regular for each EN system N . □

6. NON-SEQUENTIAL OBSERVATIONS OF EVENT OCCURRENCES AND CONDITION HOLDINGS

We will consider now a representation of the non-sequential behaviour of an EN system that is more detailed than the one obtained via firing traces. The idea is to "run" the system - resolving conflicts in an arbitrary fashion as and when they arise - and record non-sequential occurrences by events *together with* the resulting holdings of conditions during such a run. Now the records are non-sequential to start with and the resulting objects (called *processes*) are based on a special kind of nets called occurrence nets.

Definition 6.1. An *occurrence net* is a net $N = (S,T,F)$ which satisfies

(i) $(\forall s \in S)[\,|{}^\bullet s| \leq 1 \text{ and } |s^\bullet| \leq 1\,]$.

(ii) $(\forall x,y \in X_N)[\,(x,y) \in F^+ \Rightarrow (y,x) \notin F^+\,]$. □

An occurrence net is meant to model a non-sequential stretch of history and hence it is required to be acyclic. In such a history - where all conflicts have been resolved - every condition begins (ceases) to hold as the result of a unique event occurrence. Consequently every condition is required to be non-branching both at the input and output side.

Also, we do not consider events with empty output sets. *Thus for the sequel we will additionally assume that* if $N = (S,T,F)$ is an occurrence net then:

(iii) $(\forall t \in T)[\,t^\bullet \neq \emptyset\,]$.

Since an occurrence net is a bipartite graph the notion of a node-labeled occurrence net should be clear. Now we can define a process of an EN system as a node-labeled occurrence net satisfying certain "consistency requirements". We will use ${}^{\circ}N$ to denote the set $\{x \in X_N \mid {}^{\bullet}x = \emptyset\}$.

Definition 6.2. Let $N = (B,E,F,C_{in})$ be an EN system and $N = (S,T,H,\phi)$ a node-labeled occurrence net.

N is a *process of* N iff

(i) $\phi(S) \subseteq B$ and $\phi(T) \subseteq E$.

(ii) $(\forall s_1,s_2 \in S)[\phi(s_1) = \phi(s_2) \Rightarrow$ either $s_1 \leq_N s_2$ or $s_2 \leq_N s_1]$.

(iii) $(\forall t \in T)[\phi({}^{\bullet}t) = {}^{\bullet}\phi(t)$ and $\phi(t^{\bullet}) = \phi(t)^{\bullet}]$.

(iv) $\phi({}^{\circ}N) \subseteq C_{in}$. □

We use $\mathbf{P}(N)$ to denote the set of all processes of N .

It is easily seen that if N is a process, then ${}^{\bullet}t \neq \emptyset$ for every $t \in T_N$.

Example 6.1. Consider the EN system N_1 from Example 3.1.

Then the following node-labeled occurrencce net $N_1 = (S_1,T_1,H_1,\phi_1)$:

Figure 6.1

is a process of N_1 ; here we use the graphical convention that the identity of an element of N_1 (an S-element or a T-element) is indicated within a node representing it (a circle or a box respectively) while the label of the element (i.e., the value of ϕ_1) is written next to the node representing the element.

It is easily seen that the labeling ϕ_1 of the occurrence net (S_1,T_1,H_1) satisfies conditions (i) through (iv) of the above definition.

N_1 records non-sequential occurrences of events e_2, e_4, e_5, e_3 together with the holdings of appropriate conditions during a run of N_1 transforming the case $\{b_1,b_2\}$ into the case $\{b_1,b_4\}$.

Also the following occurrence net $N_2 = (S_2,T_2,H_2,\phi_2)$:

Figure 6.2

is a process of N_1. It records sequential occurrences of events (an occurrence of e_3 followed by an occurrence of e_5 followed by an occurrence of e_3) together with the holdings of appropriate conditions during a run of N_1 transforming the subcase $\{b_2\}$ (of the case $\{b_1,b_2\}$) into the subcase $\{b_4\}$ (of the case $\{b_1,b_4\}$) .

□

A useful intuition behind the notion of the process is this : if we mark the conditions in $^{\circ}N$ and play the token game on N , then this particular game should also be "permitted" by N . This intuition is formalized as follows. (A *slice* of an occurrence net N, or of a node labeled occurrence net N, is a maximal anti-chain consisting of S-elements only.)

Theorem 6.1. Let N be an EN system, $N = (S,T,F,\phi)$ a process of N and S' a slice of N. Then there exists a $C \in C_N$ such that $\phi(S') \subseteq C$. □

The notion of a process as a formalization of a behaviour of an EN system is too detailed : it turns out that two EN systems without "redundant events" with label-isomorphic sets of processes are structurally similar.

In order to formalize the notion of an EN system without "redundant events" we need the following definition.

Definition 6.3. Let N be an EN system. The set of *active events of* N, denoted ev(N), is the set $\bigcup_{U \in U_N} U$. □

It is easily seen that, in general, ev(N) may be a strict subset of E_N .

The label-isomorphism of sets of processes is understood as follows. Let W_1 be a set of node Σ_1-labeled bipartite graphs and W_2 be a set of node Σ_2-labeled bipartite graphs. We say that W_1 and W_2 are *label-isomorphic* iff there exists a function $\gamma : \Sigma_1 \longrightarrow \Sigma_2$ and a bijection $\delta : W_1 \longrightarrow W_2$ such that for each $g \in W_1$, g and $\delta(g)$ are label-isomorphic where the bijection involved between the labels of g - say Σ_g - and the labels of $\delta(g)$ is γ restricted to Σ_g .

We are ready now to formalize the above mentioned result.

Theorem 6.2. Let N_1 and N_2 be EN systems such that $ev(N_1) = E_{N_1}$ and $ev(N_2) = E_{N_2}$. Then $\mathbf{P}(N_1)$ is label-isomorphic with $\mathbf{P}(N_2)$ iff $N_1 \equiv N_2$. □

Although the process representation of the behaviour of an EN system is too detailed (as manifested by the above result) the *notion* of a process is very

useful. Indeed a theory of non-sequential processes based on occurrence nets can be built up without tying ourselves down to a system model. This theory will be discussed in the lecture by C. Fernandez.

To overcome the rather undesirable identification of an EN system with its (process) behaviour one tries to get rid of too detailed information recorded within a process. The obvious way to do it is to dispose of recording of the holdings of conditions during a run of the system.

This leads us to the following definition.

Definition 6.4. Let N be an EN system and let $\mathrm{N} = (\mathrm{S},\mathrm{T},\mathrm{F},\phi)$ be a process of N.

(i) The S-contracted version of N is a *contracted process of* N ; the set of all contracted processes of N is denoted by $\mathbf{CP}(N)$.

(ii) The *elementary event structure of* N, denoted *ees*(N), is the labeled poset $\leq_{ctr_S(\mathrm{N})}$; the set of all elementary event structures of processes of N is denoted by $\mathbf{EES}(N)$. □

By considering contracted processes, rather than processes, one disposes of the identification of an EN system with its behaviour. Now a given behaviour can be realized by (essentially) different EN systems.

Theorem 6.3. There exist EN systems N_1 and N_2 such that $\mathrm{ev}(N_1) = \mathrm{E}_{N_1}$, $\mathrm{ev}(N_2) = \mathrm{E}_{N_2}$ and $\mathbf{CP}(N_1)$ is label-isomorphic with $\mathbf{CP}(N_2)$ while it is not true that $N_1 \equiv N_2$. □

In developing the theory of behaviour of EN systems we have assumed that we deal with contact-free systems only. Now we can, somewhat informally, give (some of the) reasons for this assumption.

Consider the following EN system N_2 :

Figure 6.3

It is clear that this system can have arbitrarily long stretches of behaviour (e.g. the sequence of steps $\{e_1,e_3\}$, $\{e_2,e_4\}$ may be repeated arbitrarily many times).

On the other hand, if we would like to apply our definition of a process to N_2 then we get only a finite set of abstract processes for N_2 !

Here they are (since we consider abstract processes we indicate only the

labels of the elements of the occurrence nets) :

Figure 6.4

We note that, e.g., if we consider two possible "elementary" extensions of N_3 :

Figure 6.5

then we get node-labeled occurrence nets which are not processes of N_2 ; condition (ii) of Definition 6.2 is violated : in N_3' we have two places labeled by b_3 which are not ordered in $\leq_{N_3'}$, and in N_3'' we have two places labeled by b_1 which are not ordered in $\leq_{N_3''}$.

On the other hand if we apply the standard complement construction to N_2 we obtain the EN system N_2' :

Figure 6.6

for which the set of processes is infinite.

Here, e.g., the process "corresponding to" the node-labeled occurrence net N_3' is the following node-labeled occurrence net $\tilde{N}_3$:

Figure 6.7

Now the two places labeled by b_3 are ordered in $\leq \tilde{N}_3$!

7. COMPARING SEQUENTIAL AND NON-SEQUENTIAL APPROACHES

We have discussed two essentially different ways (methodologies) of representing the behaviour of an EN system N.

(1) Obtain the firing sequences of the system. Then use the dependence relation D_N to break these strings into dependency graphs of N to obtain **AFD**(N) and subsequently **AFLP**(N).

This methodology can be illustrated as follows:

Figure 7.1

(2) Obtain the processes of N. Then use the (S-)contraction to obtain $\mathbf{CP}(N)$ and consequently $\mathbf{EES}(N)$.

This methodology can be illustrated as follows:

Figure 7.2

The following result provides the comparison of these approaches (in what follows, $\overset{\lambda}{=}$ denotes the equality of two sets of node labeled graphs modulo the

empty graph λ) .

Theorem 7.1.

(1) There exists an EN system N such that $\mathbf{AFD}(N) \overset{\lambda}{\not=} \mathbf{CP}(N)$.

(2) For each EN system N, $\mathbf{AFLP}(N) \overset{\lambda}{=} \mathbf{EES}(N)$. □

Hence we have the following situation:

Figure 7.3

ACKNOWLEDGEMENTS

The author is indebted to IJ.J. Aalbersberg, I. Keesmaat and P.S. Thiagarajan for useful comments on the first version of theses notes, and to H. Ortiz and M. Powell for the superb typing of the manuscript in a very short time. THe author gratefully acknowledges the support by NSF Grant DCR8305245.

8. BIBLIOGRAPHICAL NOTES

The material presented here follows very closely [RT] where also the notion of an EN system was formulated.

Expressing the behaviour of Petri nets by the sets of firing sequences was the topic of very extensive investigation in 1970's. The major work that initiated this research is due to Hack [H]. Typical work on firing sequences and their generalizations is presented in [J], [JV], [P], [RV], and [S1]. Unfortunately little is known about the sets of firing sequences of EN systems (or C/E systems). The topic of firing sequences will be discussed in more depth in the lectures by M. Jantzen and R. Valk.

The theory of traces is due to A. Mazurkiewicz [M1]. An interesting application of the theory of traces to the problem of decomposition of C/E systems is presented in [M2]. A survey of the theory of traces stressing the relationship to the theory of C/E systems is presented in [AR]. The basic properties of depen-

dency graphs are studied in [ER]. The major issues of the theory of traces will be presented in the lecture by A. Mazurkiewicz.

The notion of a process (for C/E systems) was introduced by C.A. Petri in [Pt]. Typical work on the theory of processes is presented in [B], [FT], [R], and [S2]. The notion of a process for P/T systems is formulated in [GR]. The lecture by C. Fernandez will investigate many important issues in the theory of processes.

Elementary event structures were formulated in [NPW].

The relationship between the sequential and the non-sequential behaviour as discussed in our lecture (and illustrated by the final diagram of Section 7) is established in [AR] (for the case of C/E systems). Related work - mainly for P/T systems - is presented in [BF].

9. REFERENCES

[AR] IJ.J. Aalbersberg and G. Rozenberg, Theory of traces, Institute of Applied Mathematics and Computer Science, University of Leiden, Techn. Rep. No. 85-16 (1985).

[B] E. Best, A theorem on characteristics of non-sequential processes, *Fundamenta Informatica* III.1 (1980), pp. 77-94.

[BF] E. Best and C. Fernandez, Concurrent systems and processes, GMD Internal Report (1985).

[ER] A. Ehrenfeucht and G. Rozenberg, On the structure of dependency graphs, Computer Science Department, University of Colorado at Boulder, Boulder, Colorado, USA, Tech. Rep. No. CU-CS-329-86 (1986).

[FT] C. Fernandez and P.S. Thiagarajan, D-continuous causal nets: a model of non-sequential processes, *Theoretical Computer Science* 28 (1984), pp. 171-196.

[GR] U. Goltz and W. Reisig, The non-sequential behaviour of Petri nets, *Information and Control* 57 (1983), pp. 125-147.

[H] M.H.T. Hack, Petri net languages, Computation Structures Group Memo 124, Project MAC, M.I.T., Cambridge, Massachusetts, USA (1976).

[J] M. Jantzen, On the hierarchy of Petri net languages, *RAIRO Theoretical Informatics* 19 (1979), pp. 19-30.

[S1] P. Starke, Free Petri net languages, *Lecture Notes in Computer Science* 64 (1978), pp. 506-515.

[S2] P. Starke, Processes in Petri net, *Electronische Informationsverarbeitung und Kybernetik* 17 (1981).

[JV] M. Jantzen and R. Valk, Formal properties of Place/Transition systems, *Lecture Notes in Computer Science* 84 (1980).

[M1] A. Mazurkiewicz, Concurrent program schemes and their interpretation, Computer Science Deparatment, Aarhus University, Aarhus, Denmark, Techn. Rep. No. PB-78 (1978).

[M2] A. Mazurkiewicz, Semantics of concurrent systems: a modular fixed-point trace approach, *Lecture Notes in Computer Science* 188 (1984), pp. 353-375.

[NPW] M. Nielsen, G. Plotkin and G. Winskel, Petri nets, event structures and domains, Part I, *Theoretical Computer Science* 13 (1981), pp. 85-108.

[P] J.L. Peterson, Computation sequence sets, *Journal of Computer and System Sciences* 13 (1976), pp. 1-24.

[Pt] C.A. Petri, Non-sequential processes, GMD, St. Augustin, W. Germany, Internal Report GMD-ISF-77.5 (1977).

[R] W. Reisig, *Petri nets: An Introduction,* EATCS Monographs on Theoretical Computer Science, Springer-Verlag, Heidelberg (1985).

[RT] G. Rozenberg and P.S. Thiagarajan, Petri nets: basic notions, structure, behaviour, *Lecture Notes in Computer Science* 224 (1986), pp. 585-668.

[RV] G. Rozenberg and R. Verraedt, Subsets languages of Petri nets, Part I, *Theoretical Computer Science* 26 (1983), pp. 301-326.