Bull. Nov. Comp. Center, Comp. Science, 3 (1995), 101–115

Modelling the semantics of coloured dataflow networks*

I.B. Virbitskaite and A.V. Votintseva

1. Introduction

Dataflow networks are a well-known mathematical tool extensively used for modelling and analyzing concurrent computing systems and their software. A few formal dataflow models reported in the literature may be amalgamated in two groups–static and dynamic. Static models [4] allow at most one token on an arc. This assumption severely limits the amount of concurrency that is possible. Dynamic models are free from this restriction by program code copying [11] and token colouring [2, 12].

There exist various approaches to represent the behaviour of dataflow networks in terms of different notions allowing concurrency to be naturally expressed. Modelling the operational semantics of static dataflow networks by terms of the process algebra ACP was presented in [3]. Modularity and Kahn's principle were investigated in [9, 10] for dataflow networks whose semantics was represented by pomsets and trace languages. A fully abstract trace-model for dataflow networks was given in [5].

Our aim here is to develop a number of behaviour notions (firing sequences, trace languages, dependence graphs and event structures) for coloured dataflow networks and establish formal relationships between these notions. The paper is organized as follows. In Section 2, we introduce the notion of a coloured (dataflow) network and define its semantics in terms of firing sequences. In Section 3, the trace semantics of a coloured network is given. In Section 4, to each firing sequence we associate a dependence graph that is an other representation of the semantics of a coloured network. There it is further established that for a coloured network its trace language agrees in a strong way with the set of its dependence graphs. In Section 5, the basic notions and observations concerning event structures are presented. There it is further shown how to get the event structure semantics of a coloured network from its dependence graphs. The section

*Partially supported by INTAS (International Association for the Promotion of Cooperation with Scientists from the Independent States of the Former Soviet Union), Contract No. 1010–CT93–0048.

concludes with the comparison of the mentioned above semantics. In final section some concluding remarks are given.

2. Coloured dataflow networks

A coloured (dataflow) network is characterized by nodes, arcs and a distribution of coloured tokens. The nodes consist of links and actors. There are two kinds of link nodes (data and control links) and four kinds of actor nodes (operators, deciders, gates and colour actors). The arcs connecting links with actors and actors with links are called data and control arcs according to the type of link. Tokens flowing through arcs carry values between nodes of the network. We distinguish data tokens (carrying arbitrary values from the domain of an interpretation for a network) and control tokens (carrying truth values (**true**, **false**)). Moreover, each token has some colour. For more information about informal description of coloured networks see [13].

Now we formally define coloured networks. Some explanations given below are a part of the formal definition. It is recommended to read these explanations in parallel with the main definition.

Definition 2.1. A *coloured network* is a quadruple $\mathcal{N} = (N, E, T, C)$, where

(i) N is a set of nodes consisting of a subset L of links and a subset A of actors. The links are of two types: data links (L^D) and control links (L^R). The actors are of the following types: operators (A^F), deciders (A^R), gates (A^G) and colour actors ($A^C = \mathsf{New} \cup \mathsf{Next} \cup \mathsf{Old}$).

(ii) $E \subseteq (A \times (L \cup \omega)) \cup ((L \cup \omega) \times A)$ ($\omega \notin N$) is a set of arcs consisting of a subset E^D of data arcs and a subset E^R of control arcs. $\omega \notin N$.

(iii) T is a set of tokens consisting of a subset T^D of data tokens and a subset T^R of control tokens.

(iv) $C : T \longrightarrow \Sigma$ is a colour function.

(i), (ii): the nodes and arcs are represented by two sets N and E which have to be finite and disjoint. The arcs are of two types: data arcs (E^D) and control arcs (E^R). The set of nodes consists of two disjoint subsets L (links) and A (actors). Figure 1a) shows the types of links. At least one data (control) arc must terminate on and at least one data (control) arc must originate at each data (control) link. The types of actors are shown in Figure 1b).

1. Operator $f \in A^F$. An operator has an ordered set of input data arcs and a single output data arc.

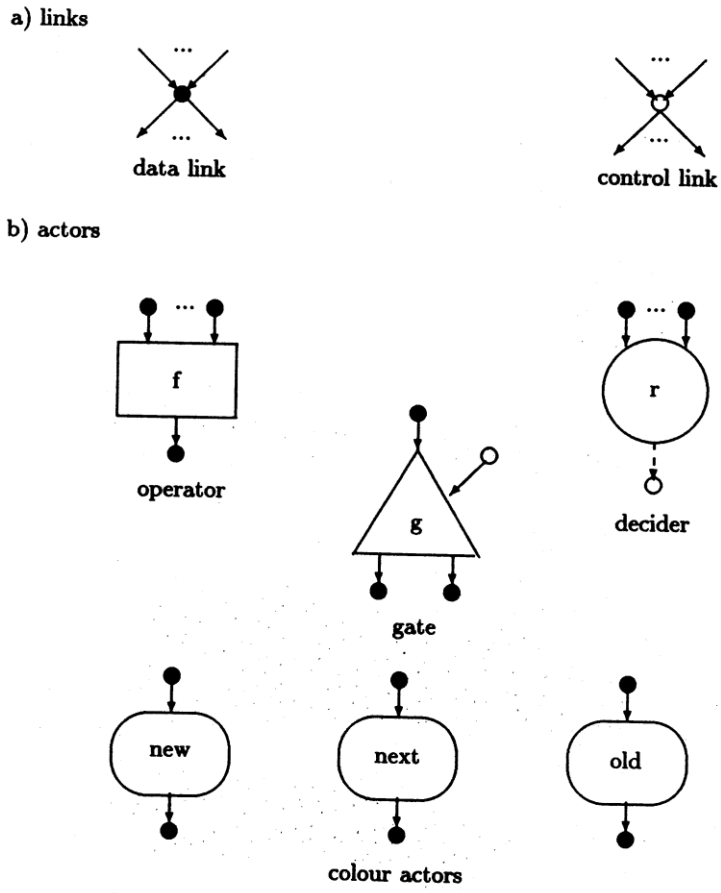

Figure 1

2. Decider $r \in A^R$. A decider has an ordered set of input data arcs and a single output control arc.
3. Gate $g \in A^G$. A gate has one input data arc and one input control arc and two output data arcs, labelled by '+'- and '–'-signs.
4. Colour actors: *new* $\in$ *New*, *next* $\in$ *Next*, *old* $\in$ *Old*, allowing loop computations to be modelled. A colour actor has a single input data arc and a single output data arc.

For a node $n \in N$, let $in(n)$ and $out(n)$ denote the set of its input arcs and the set of its output arcs, respectively. Let $\mathrm{In} = \{(\omega, n) \in E \mid \omega \notin N \;\&\; n \in A\}$ (the set of input arcs of $\mathcal{N}$) and $Out = \{(n, \omega) \in E \mid \omega \notin N \;\&\; n \in A\}$ (the set of output arcs of $\mathcal{N}$).

(iii): the tokens are defined by two disjoint subsets T^D (data tokens) and T^R (control tokens).

(iv): the colour function C maps T on a finite non-empty set Σ of colours. Each colour is a triple $c = (x, y, z)$, where $c.x$ is the name of the loop, $c.y$ is the number of the loop iteration, and $c.z$ is the context that may be itself a colour. (If a token occurs outside a loop, its colour is $(0, 0, z)$ with $c.z \in \mathbf{N}$, where $\mathbf{N}$ is the set of natural numbers.)

Figure 2 is an example of a coloured network.

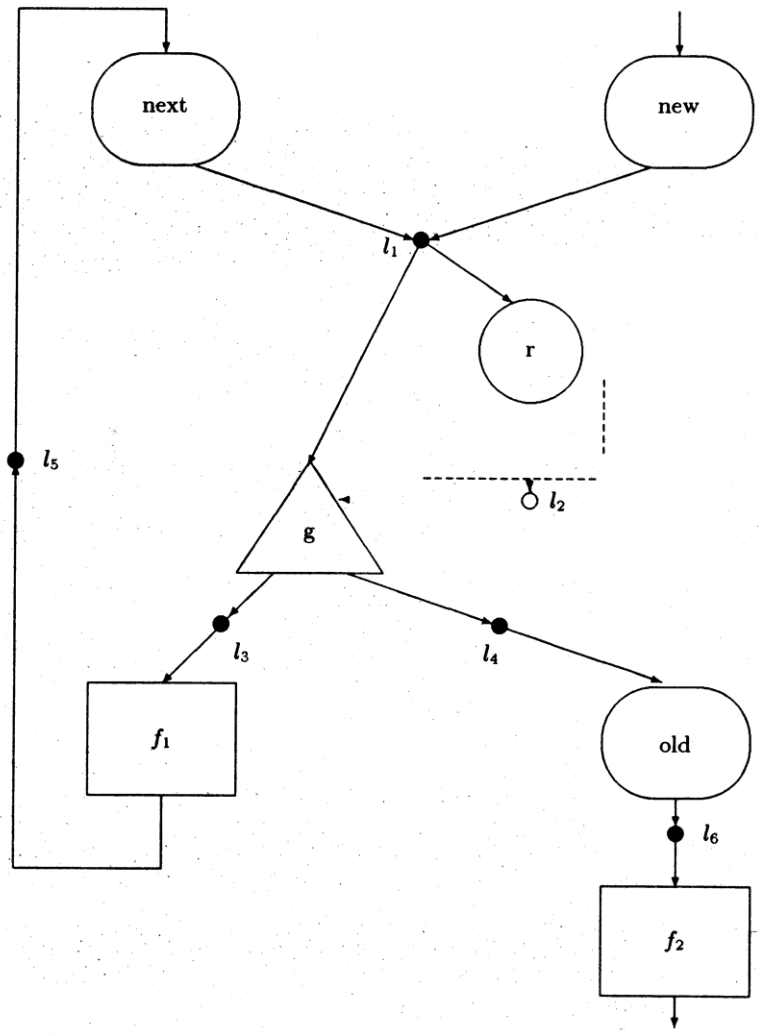

Figure 2

Let $n, n' \in N$. Then ${}^\bullet n = \{n' \mid n' \; E \; n\}$ (the set of input elements of n) and $n^\bullet = \{n' \mid n \; E \; n'\}$ (the set of output elements of n).

A coloured network $\mathcal{N}$ is an uninterpreted model. Specifying an interpretation for $\mathcal{N}$ provides a complete representation of computation.

Definition 2.2. Let $\mathcal{N}$ be a coloured network. Then an *interpretation* I for $\mathcal{N}$ is defined as follows:

(i) A domain $\mathbf{D}$ of values,

(ii) an assignment of a total function $\varphi : \mathbf{D}^q \longrightarrow \mathbf{D}$ to each operator $f \in A^F$, where $q = |\operatorname{in}(f)|$,

(iii) an assignment of a total predicate $\pi : \mathbf{D}^q \longrightarrow \{\mathbf{true}, \mathbf{false}\}$ to each decider $r \in A^R$, where $q = |\operatorname{in}(r)|$.

Introduce the value function V that assigns a value $V(t) \in \mathbf{D}$ ($V(t) \in \{\mathbf{true}, \mathbf{false}\}$, respectively) to each token $t \in T^D$ ($t \in T^R$, respectively). We use a pair $(\mathcal{N}, I)$ to denote the interpreted coloured network. Let $\mathbf{I}$ denote the set of all possible interpretations of $\mathcal{N}$.

A *state* in $(\mathcal{N}, I)$ is a function S defined from A into 2^T such that the tokens must have a type identical to the type of the arc. By $\mathbf{S}(\mathcal{N}, I)$ we denote the set of all states in $(\mathcal{N}, I)$, and let $\mathbf{S} = \cup(\mathbf{S}(\mathcal{N}, I) \mid I \in \mathbf{I})$. A state S in $(\mathcal{N}, I)$ is *input* (S_{in}), iff $(\forall e \in \text{In}\ \forall t, t' \in S(e)\ .\ C(t) \neq C(t')) \ \&\ (\forall e \in (E \setminus In)\ .\ S(e) = \emptyset)$. From now on let S and S' range over $\mathbf{S}(\mathcal{N}, I)$.

The *firing rule* associated with a node $n \in N$ is defined as follows: There is a colour $c = (x, y, z) \in \Sigma$ such that

(i) An actor node (link node) n is *enabled* with a colour c in a state S if there is a token with a colour c on each (at least one) input arc of n.

(ii) A node n enabled with a colour c in a state S may be choosen to fire with a colour c yielding a new state S' specified as follows:

(a) One token with a colour c is removed from each (at most one) input arc of an actor node (link node) n.

(b) The tokens are added to the output arcs of n in the following way:

2.1. For a data (control) link node n, one data (control) token with a colour c and a value $v = V(t)$ is added to each output arc of n, where t is the data (control) token removed from the input arc of n.

2.2. For an operator (decider) node n, one data (control) token with a colour c and a value $v = \varphi(V(t_1), \ldots, V(t_q))$ ($v = \psi(V(t_1), \ldots,$ $V(t_q))$) is added to the output arc of n, where $t_1, \ldots, t_q$ are the data tokens removed from the input arcs of n and $q = |\operatorname{in}(n)|$.

2.3. For a gate node n, one data token with a colour c and a value $v = V(t_1)$ is added to the '+'-port if $V(t_2) = \mathbf{true}$ or to the

'–'-port if $V(t_2) = \mathbf{false}$, where t_1 and t_2 are respectively the data and control tokens removed from the input ports of n.

2.4. For a colour actor node n, one data token with a colour $c' = U(c)$ and a value $v = V(t)$ is added to the output port of n, where t is the data token removed from the input port of n and

$$U(c) = \begin{cases} (new, 0, c), & \text{if } n \in New, \\ (c.x, c.y + 1, c.z), & \text{if } n \in Next, \\ c.z, & \text{if } n \in Old. \end{cases}$$

A *firing* with a colour c of a node n is a triple $S \xrightarrow{(n,c)} S'$ such that a transition from the state S to the state S' is consistent with the firing rule of n.

We can now introduce the first and the most primitive semantic representation of a coloured network. A *firing sequence* in $(\mathcal{N}, I)$ is a string ρ over alphabet $(N \times \Sigma)$ defined as:

(i) $\rho = \Lambda$ and $S_{in} \xrightarrow{\Lambda} S_{\text{in}}$,

(ii) Suppose ρ' is a firing sequence in $(\mathcal{N}, I)$, $S_{\text{in}} \xrightarrow{\rho'} S$ and $S \xrightarrow{(n,c)} S'$, then $\rho = \rho'(n, c)$ and $S_{\text{in}} \xrightarrow{\rho'(n,c)} S'$.

Let $\mathbf{R}(\mathcal{N}, I)$ denote the set of all firing sequences in $(\mathcal{N}, I)$, and let $\mathbf{R} = \cup(\mathbf{R}(\mathcal{N}, I) \mid I \in \mathbf{I})$. We will say that a firing sequence ρ in $(\mathcal{N}, I)$ is *safe* iff $\forall (n, c), (n', c') \in \rho \,.\, (n, n' \in A \;\&\; n^\bullet \cap n'^\bullet \neq \emptyset) \Longrightarrow c \neq c'$.

We are ready to begin our study of different semantic representations of a coloured network. For the sake of convenience we fix a coloured network $\mathcal{N} = (N, E, T, C)$ and work with it throughout what follows. We shall assume that $\mathcal{N}$ is *behaviourally safe*. In other words, we shall assume, $\forall I \in \mathbf{I}\ \forall \rho \in \mathbf{R}(\mathcal{N}, I)$. ρ is safe.

Clearly, firing sequences hide information concerning concurrency and nondeterministic choices (conflicts). We will now see how the theory of trace languages can be applied to extract such information.

3. The trace semantics of $\mathcal{N}$

The theory of trace languages was proposed to model the nonsequential behaviour of distributed programs. The interested reader is referred to [1, 6] for details. Here we shall straight away apply the notions of this formalism to coloured networks. Before doing so, we need to introduce some additional notations.

Through the rest of this section we set $R = (N \times \Sigma)$ and $R' = ((A^F \cup A^n) \times \Sigma)$. For $\rho \in \mathbf{R}$, σ_ρ will denote the projection of ρ onto R', i.e., a string over $R \cap R'$ defined as:

(i) $\sigma_\Lambda = \Lambda$,

(ii) $\sigma_{\rho(n,c)} = \begin{cases} \sigma_\rho(n,c), & \text{if } (n,c) \in R', \\ \sigma_\rho, & \text{otherwise.} \end{cases}$

For two actor nodes n, n', we will write $n \hookrightarrow n'$, iff there exists $l \in L$ such that $(n \in^\bullet l)$ & $(n' \in l^\bullet)$. We use $n \overset{*}{\hookrightarrow} n'$ to denote the following fact: $n \hookrightarrow n_1$ & ... & $n_m \hookrightarrow n'$, where $n, n' \in A$, $n_1, \ldots, n_m \in A \setminus (A^F \cup A^R)$ and $m \geq 0$.

By $\mathcal{D} \subseteq R' \times R'$ we denote the dependence relation associated with $\mathcal{N}$, $\mathcal{D} = \{((n,c),(n',c')) \mid (n \overset{*}{\hookrightarrow} n' \vee n' \overset{*}{\hookrightarrow} n)$ & $(c = c')\}$. Define $\mathcal{I} = R'^2 \setminus \mathcal{D}$, $\mathcal{I}$ will be called the *independence* relation associated with $\mathcal{N}$. Obviously, $\mathcal{I} \subseteq R' \times R'$ is a reflexive and symmetric relation. Then we have a natural way of partitioning R' using the least congruence relation generated by $\mathcal{I}$ via equations of the form $(n,c)(n',c') = (n',c')(n,c)$, where $((n,c),(n',c')) \in \mathcal{I}$.

Define $\dot{\sim} \subseteq (R \cap R')^*$ as follows:

$$\sigma_\rho \dot{\sim} \sigma_{\rho'} \overset{\text{def}}{\Longleftrightarrow} \exists \rho_1, \rho_2 \in R^* \; \exists n, n' \in ID \mid \sigma_\rho = \sigma_{\rho_1}(n,c)(n',c')\sigma_{\rho_2} \;\&\; \sigma_{\rho'} = \sigma_{\rho_1}(n',c')(n,c)\sigma_{\rho_2}.$$

Then $\sim = (\dot{\sim})^*$ is the equivalence relation we want and for $\rho \in \mathbf{R}$, $[\sigma_\rho] \overset{\text{def}}{=} \{\sigma_{\rho'} \mid \rho' \in \mathbf{R}$ & $\sigma_\rho \sim \sigma_{\rho'}\}$.

Let $\mathbf{T} \overset{\text{def}}{=} \{[\sigma_\rho] \mid \rho \in \mathbf{R}\}$. Now we need to introduce an ordering relation over $\mathbf{T}$. $\sqsubseteq \subseteq \mathbf{T} \times \mathbf{T}$ is given by:

$$t \sqsubseteq t' \overset{\text{def}}{\Longleftrightarrow} \forall \sigma_\rho \in t, \sigma_{\rho'} \in t' \,.\, \sigma_\rho \in \text{Prefix}(\sigma_{\rho'}).$$

Here Prefix(γ) denotes the set of prefixes of the string γ. It is easy to check that $(\mathbf{T}, \sqsubseteq)$ is a poset. Figure 3 shows an initial portion of the poset of traces associated with the coloured network shown in Figure 2.

To see the information concerning nondeterministic choice we define a 'compatibility' relation over $\mathbf{T}$ as follows. Let $t, t' \in \mathbf{T}$. Then

$$t \uparrow t' \overset{\text{def}}{\Longleftrightarrow} \exists t'' \in \mathbf{T} \,.\, t \sqsubseteq t'' \;\&\; t' \sqsubseteq t''.$$
$$t \not\uparrow t' \overset{\text{def}}{\Longleftrightarrow} \neg(t \uparrow t').$$

In the example shown in Figure 3 $(r,c_1)(f_2,c_0) \not\uparrow (r,c_1)(f_1,c_1)$, since the choice of (f_2,c_0) in the firing sequence (new,c_0) (l_1,c_1) (r,c_1) (l_2,c_1) (g,c_1) (l_4,c_1) (old,c_1) (l_6,c_0) (f_2,c_0) is oposed to the choice of (f_1,c_1). in the firing sequence (new,c_0) (l_1,c_1) (r,c_1) (l_2,c_1) (g,c_1) (l_3,c_1) (f_1,c_1). Here $c_0 = (0,0,0)$, $c_1 = (new,0,c_0)$.

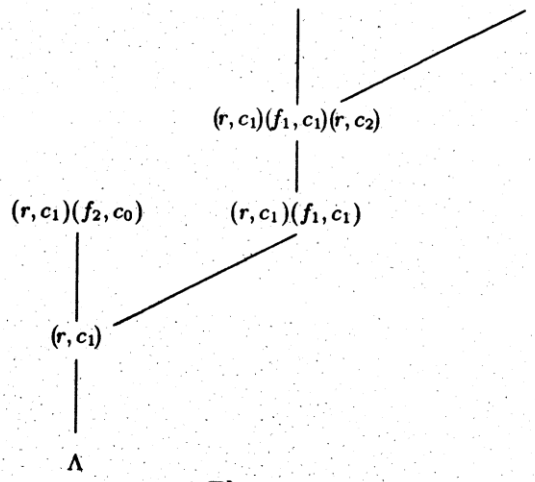

Figure 3

4. The dependence graphs of $\mathcal{N}$

Now we wish to find a next semantic representation of a coloured network. This representation will be in terms of *dependence graphs* (d-graphs) which are a variant of a traditional semantic notion – data dependence graphs [4]. We build up the d-graphs of a coloured network inductively with the help of its firing sequences. As we will see, this method of constructing d-graphs will be very helpful for getting the desired results.

To each firing sequence ρ we will associate a d-graph $G_\rho = (\hat{V}_\rho, \hat{E}_\rho)$ as follows.

Definition 4.1. Let $\rho \in \mathbf{R}$. Then $G_\rho = (\hat{V}_\rho, \hat{E}_\rho)$ is given by:

$\rho = \Lambda$. Then $G_\Lambda = (\emptyset, \emptyset)$, and $K_\Lambda = \emptyset$.

$\rho \neq \Lambda$. Let $\rho = \rho'(n, c)$ and assume that $G_{\rho'} = (\hat{V}_{\rho'}, \hat{E}_{\rho'})$ and $K_{\rho'}$ are defined. Then $G_\rho = (\hat{V}_\rho, \hat{E}_\rho)$ with

$$\hat{V}_\rho = \begin{cases} \hat{V}_{\rho'} \cup \{(n, c)\}, & \text{if } n \in A^F \cup A^R, \\ \hat{V}_{\rho'}, & \text{otherwise,} \end{cases}$$

$$\hat{E}_\rho = \begin{cases} \hat{E}_{\rho'} \cup (K_{\rho'}(n, c) \times \{(n, c)\}, & \text{if } n \in A^F \cup A^R, \\ \hat{E}_{\rho'}, & \text{otherwise,} \end{cases}$$

K_ρ is defined by:

$$\forall\, (n', c') \in \hat{V}_{\rho'} : \; K_\rho((n', c')) = K_{\rho'}((n', c'))$$

and

$$K_\rho((n,c)) = \begin{cases} \{(n,c)\}, & \text{if } n \in A^F \cup A^R, \\ \cup(K_{\rho'}((n',c')) \mid \text{out}(n') \cap \text{in}(n) \neq \emptyset \ \& \ c' \text{ is as below}), \\ \qquad \text{otherwise}, \end{cases}$$

$$c' = \begin{cases} U^{-1}(c), & \text{if} \quad n' \in A^C, \\ c, & \text{otherwise.} \end{cases}$$

Let $\mathbf{G} = \{G_\rho \mid \rho \in \mathbf{R}\}$ denote the set of finite d-graphs of $\mathcal{N}$. Figure 4 shows the d-graph associated with the firing sequence (new, c_0) (l_1, c_1) (r, c_1) (l_2, c_1) (g, c_1) (l_3, c_1) (f_1, c_1) (l_4, c_1) (next, c_1) (l_1, c_2) (r, c_2) (l_2, c_2) (g, c_2) (l_3, c_2) (f_1, c_2) (l_4, c_2) (next, c_2) (l_1, c_3) (r, c_3) (l_2, c_3) (g, c_3) (l_5, c_3) (old, c_3) (l_6, c_0) (f_2, c_0), where $c_0 = (0,0,0)$, $c_1 = (\mathsf{new}, 0, c_0)$, $c_2 = (\mathsf{new}, 1, c_0)$, $c_3 = (\mathsf{new}, 2, c_0)$ (see the coloured network in Figure 2).

$$(f_1, c_1) \longrightarrow (f_1, c_2) \longrightarrow (f_2, c_0) \longrightarrow (r, c_1) \longrightarrow (r, c_2) \longrightarrow (r, c_3)$$

Figure 4

In order to establish a relationship between the traces and the d-graphs of a coloured network it is necessary to define an ordering relation over $\mathbf{G}$. Let $\subseteq' \subseteq \mathbf{G} \times \mathbf{G}$ be defined as:

$$G_\rho = (\hat{V}_\rho, \hat{E}_\rho) \subseteq' G_{\rho'} = (\hat{V}_{\rho'}, \hat{E}_{\rho'}) \overset{\text{def}}{\Longleftrightarrow} \hat{V}_\rho \subseteq \hat{V}_{\rho'} \quad \text{and} \quad \hat{E}_\rho \subseteq \hat{E}_{\rho'}.$$

Clearly, $\subseteq'$ is a partial ordering relation.

Proposition 4.2. $(\mathbf{T}, \sqsubseteq)$ *and* $(\mathbf{G}, \subseteq')$ *are isomorphic posets. In fact,* $h : \mathbf{T} \longrightarrow \mathbf{G}$ *given by* $\forall \rho \in \mathbf{R} \ . \ h([\sigma_\rho]) = G_\rho$ *is an isomorphism.*

5. The event structure semantics of $\mathcal{N}$

Now we recall some terminology concerning event structures (prime event structures introduced in [7]) denoting the behaviour of systems. Event structures are represented via sets of events with relations expressing causal dependencies and conflicts between them. The subsets of events representing executions in the event structure are called configurations. They have to be conflict-free and left-closed with respect to $\leq$ (all prerequisites for any event occurring in the execution must also occur).

Definition 5.1. *An event structure* is a triple $\mathcal{E} = (E, \leq, \#)$, where

- E is a set of events;
- $\leq \subseteq E \times E$ is a partial order (*the causality relation*), satisfying the principle of finite causes:

$$\forall e \in E : \{d \in E \mid d \leq e\} \text{ is finite;}$$

- $\# \subseteq E \times E$ is a symmetric and irreflexive relation (*the conflict relation*) satisfying the principle of conflict heredity:

$$\forall e_1, e_2, e_3 \in E \,.\, e_1 \leq e_2 \;\&\; e_1 \# e_3 \Rightarrow e_2 \# e_3.$$

For $\mathcal{E} = (E, \leq, \#)$, let: $id = \{(e,e) \mid e \in E\}$; $< \,= \leq \setminus id$; $<^2 \subseteq <$ (transitivity); $\lhd \,= < \setminus <^2$; $e \#_1 d \iff e \# d \;\&\; \forall e_1, d_1 \in E : (e_1 \leq e \;\&\; d_1 \leq d) \Rightarrow (e_1 = e \;\&\; d_1 = d)$ (immediate conflict).

A configuration of an event structure $\mathcal{E} = (E, \leq, \#)$ is a subset C of E such that:

(i) $\forall e, e' \in C : \neg(e \# e')$ (conflict-free),

(ii) $\forall e, e' \in E \,.\, e \in X \;\&\; e' \leq e \Rightarrow e' \in C$ (left-closed).

We shall denote by $\mathbf{C}(\mathcal{E})$ the set of finite configurations (i.e., each member of $\mathbf{C}(\mathcal{E})$ is a finite set) of $\mathcal{E}$.

In a graphic representation only immediate conflicts – not the inherited ones – are pictured. The $\lhd$-relation is represented by arcs, omitting those derivable by transitivity. Following these conventions, the example of a graphical representation of the event structure is shown in Figure 5. Possible configurations of this structure are: $\emptyset$, $\{e_1\}$, $\{e_4\}$, $\{e_1, e_3\}$, $\{e_1, e_2\}$, $\{e_1, e_4\}$, $\{e_1, e_2, e_4\}$.

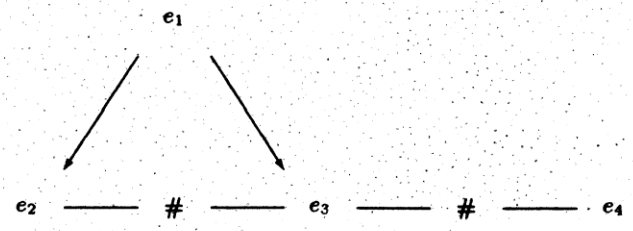

Figure 5

Now we can give the event structure semantics of $\mathcal{N}$.

Definition 5.2. The event structure of $\mathcal{N}$ is the triple $\mathcal{E}(\mathcal{N}) = (E, \leq, \#)$, where (recall that $G_\rho = (\hat{V}_\rho, \hat{E}_\rho)$ for each $\rho \in \mathbf{R}$ as specified in Definition 3.1)

- $E = \cup_{\rho \in \mathbf{R}} \hat{V}_\rho$,
- $\leq = \cup_{\rho \in \mathbf{R}} (\hat{E}_\rho^*)$,
- $\forall (n,c), (n',c') \in E \,.\, (n,c) \# (n',c') \iff$
  $\forall \rho \in \mathbf{R} \,.\, \{(n,c), (n',c')\} \not\subseteq \hat{V}_\rho$.

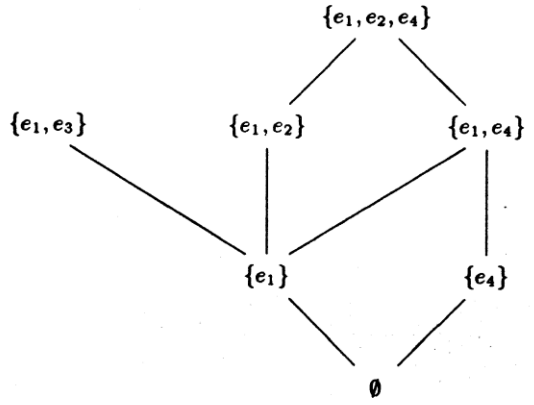

Figure 6

Figure 6 shows the event structure of the coloured network in Figure 2. For $\mathcal{E}(\mathcal{N})$, let $\mathbf{C}$ denote the set of all finite configurations of $\mathcal{E}(\mathcal{N})$.

Proposition 5.3. *$(\mathbf{G}, \subseteq')$ and $(\mathbf{C}, \subseteq)$ are isomorphic posets. In fact, $h' : \mathbf{G} \longrightarrow \mathbf{C}$ given by $\forall \rho \in \mathbf{R} \,.\, h'(G_\rho = (\hat{V}_\rho, \hat{E}_\rho)) = \hat{V}_\rho$ is an isomorphism.*

It will be convenient to recall some notions concerning posets. Let $PO = (D, \leq)$ be a poset. Then for $D' \subseteq D$, $\sqcup D'$ will denote the l.u.b. of D in PO if it exists. Two elements x and y of D are *compatible*, in notation $x \uparrow y$, if they have an upper bound, that is: $x \uparrow y \Leftrightarrow \exists z \in D \mid x \leq z \;\&\; y \leq z$. A subset D' of D is *pairwise consistent*, in notation $D' \Uparrow$, if every two elements of D' are compatible in D, that is: $D' \Uparrow \Leftrightarrow \forall x, y \in D' \,.\, x \Uparrow y$. PO is said to be *coherent*, if every pairwise consistent subset D' of D has a l.u.b. in PO. PO is said to be *finitary coherent*, if every finite pairwise consistent subset D' of D has a l.u.b. in PO. $x \in D$ is called a *prime* element, if for every $D' \subseteq D$ such that $\sqcup D'$ exists $x \leq \sqcup D'$ implies that $x \leq y$ for some $y \in D'$. Let $PR(PO)$ denote the set of prime elements of PO. PO is said to be *prime algebraic* iff $\forall x \in D \,.\, x = \sqcup\{y \mid y \in PR(PO) \;\&\; y \leq x\}$. $x \in D$ is called *finite* element, if it only dominates a finite number of elements, that is, if the set $\{y \mid y \leq x\}$ is finite. We shall denote the set of finite elements of PO by $FI(PO)$. PO is *finitary* iff $PR(PO) \subseteq FI(PO)$.

We can now state the announced result:

Proposition 5.4 [8]. *Let $\mathcal{E}$ be an event structure. Then $(\mathbf{C}(\mathcal{E}), \subseteq)$ is a prime algebraic, finitary coherent and finitary poset.*

Figure 7 shows the poset of configurations of the event structure in Figure 5.

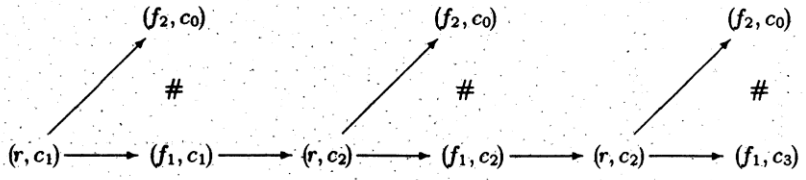

Figure 7

Proposition 5.5.

(i) $(\mathbf{T}, \sqsubseteq)$ *and* $(\mathbf{C}, \subseteq)$ *are isomorphic posets,*

(ii) $(\mathbf{T}, \sqsubseteq)$ *and* $(\mathbf{G}, \subseteq')$ *are prime algebraic, finitary coherent and finitary posets.*

6. Concluding remarks

In this paper we have formalized a number of behavior notions (firing sequences, trace languages, dependence graphs and event structures) of coloured (dataflow) networks. We have proved strong formal relationships between the above notions. In a certain sense our results are related to and extend the well-known results concerning behavior notions of elementary net systems [8]. It is worth remarking that the obtained results were formulated in terms of finite objects (finite dependence graphs and the poset of finite configurations of event structures) since the trace language we have defined contains only finite strings. Fortunately this restriction involves no permanent loss of information concerning infinite behaviours.

References

[1] I.J. Aalbersberg, G. Rozenberg, *Theory of Traces*, Tech. Report 86–16, University of Leiden, Leiden, Netherlands, 1986.

[2] Arvind, K.P. Gostelow, *U-interpretator*, Computer, Vol. 15, No. 2, 1982, 42–49.

[3] J.A. Bergstra, J.W. Klop, *A process algebra for the operational semantics of static data flow networks*, Tech. Report IW 222/83, CWI, Amsterdam, 1983.

[4] J.B. Dennis, *Data flow schemata*, Lecture Notes in Comput. Sci., Vol. 5, 1972, 187–216.

[5] B.A. Jonson, *A fully abstract trace model for dataflow networks*, POPL'89, 1989, 155–165.

[6] A. Mazurkiewicz, *Basic notions of trace theory*, Lecture Notes in Comput. Sci., Vol. 354, 1988, 285–363.

[7] M. Nielsen, G. Plotkin G, G. Winskel, *Petri nets, event structures and domains*, Theoret. Comput. Sci., Vol. 13, No. 1, 1981, 85–108.

[8] M. Nielsen, G. Rozenberg, P.S. Thiagarajan, *Behavioural notions for elementary net systems*, Distrib. Computing, Vol. 4, No. 1, 1990, 45–57.

[9] A. Rabinovich, *Pomset semantics is consistent with data flow semantics*, Bulletin of EATCS, Vol. 32, 1987.

[10] A. Rabinovich, B.A. Trakhtenbrot, *Nets of processes and data flow*, Lecture Notes in Comput. Sci., Vol. 354, 1988, 574–602.

[11] J. Rumbaugh, *A paprllel asynchronous computer architecture for data flow programs*, Tech. Report TR-150, MIT, Project MAC, May 1975.

[12] A.H. Veen, *A formal model for data flow programs with token coloring*, Tech. Report IW 179, CWI, Amsterdam, 1981.

[13] I.B. Virbitskaite, A.V. Votintseva, *Semantic models of dataflow computing*, Tech. Report 27, IIS, Novosibirsk, 1993 (in Russian).