View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Elsevier - Publisher Connector

Theoretical Computer Science 113 (1993) 231–258
Elsevier

A true concurrency model of CCS semantics

Lu Ruqian

Institute of Mathematics, Academia Sinica, 100080 Beijing, People's Republic of China

Communicated by M. Nivat
Received August 1991
Revised November 1991

Abstract

Ruqian, L., A true concurrency model of CCS semantics, Theoretical Computer Science 113 (1993) 231–258.

Degano et al. (1989) introduced AC/E systems (augmented C/E systems) to give a true concurrency semantics to CCS. But the true concurrency was not complete. There was no true concurrency for recursive agents (like $\langle x\rangle.e_1|e_2$) and nondeterminant agents (like $e_1|e_2+e_3|e_4$). Also the concept of bisimulation has not been transplanted to AC/E systems. This paper defines a complete true concurrency model of CCS by exploiting the potential concurrency of any CCS agent to its full strength. It introduces a kind of multilayered Petri nets, called NP/R nets, to define the processes on AC/E systems. We also introduced the notion of bisimulation of groups of NP/R nets and proved that this bisimulation relation can determine the CCS bisimulation uniquely.

1. Introduction

The interleaving semantics of CCS given by Milner [3] is well-known. This semantics allows the actions of concurrent agents to occur in different orders, but not "without orders". Therefore, the interleaving semantics is not considered as a true concurrent one. Degano et al. [1] used a new kind of net systems, the so-called contact-free augmented C/E systems (AC/E systems), to give a true concurrent semantics to CCS. They decomposed a CCS agent into a set of *grapes* which can act concurrently. The derivation of a set of grapes corresponds to the firing of an event of the related AC/E system. Therefore, the concurrency of event firings of AC/E systems guarantees the concurrency of grapes derivations and thus the concurrency of CCS

Correspondence to: Lu Ruqian, Institute of Mathematics, Academia Sinica, 100080 Beijing, People's Republic of China.

0304-3975/93/$06.00 © 1993—Elsevier Science Publishers B.V. All rights reserved

agent derivations. But their solution was not perfect. According to their approach, the first step of the derivation of $e_1|e_2+e_3|e_4$ or $\langle x\rangle.e_1|e_2$ (recursive agent) can only be sequential, not concurrent. Another defect of their approach was the absence of bisimulation concepts for AC/E systems, the importance of which should not be underestimated. Without this concept one cannot compare the behaviors of two AC/E systems and their relation to behaviors of CCS agents. The reason of this absence may be the fact that the AC/E systems defined by Degano et al. are in general not contact-free in the usual net-theoretical sense. It lacks a tool (like the occurrence net) to define processes of such systems. In this paper, we introduce a new kind of Petri nets, called NP/R nets, to define processes on AC/E systems. We define also the bisimulation concept for groups of NP/R nets and prove that this bisimulation relation is necessary and sufficient for the existence of a bisimulation relation between corresponding CCS agents. We also solve the problem of completely decomposing CCS agents into sets of concurrent grapes, including those of form $e_1|e_2+e_3|e_4$ or $\langle x\rangle.e_1|e_2$. Thus, we give a new true concurrent semantics to CCS.

2. AC/E systems and NP/R nets

Definition 2.1. Let $N=(S,T,F)$ be a net, M be a subset of S, called a marking. An event $e\in T$ is said to be *a-enabled* under the marking M, if the following condition is fulfilled:

$$^\bullet e\subseteq M \;\;\&\;\; e^\bullet\subseteq {}^\bullet e\cup(S-M). \tag{1}$$

Definition 2.2. Let $N=(S,T,F)$ be a net, $M_1, M_2\subseteq S, B\subseteq T$, and suppose that the following conditions are fulfilled:

(a) $\forall e\in B$, e is a-enabled under M_1,

(b) $\forall e_1,e_2\in B,\ e_1\neq e_2 \rightarrow {}^\bullet e_1\cap{}^\bullet e_2=e_1^\bullet\cap e_2^\bullet=\emptyset,$ (2)

(c) $M_2=(M_1-{}^\bullet B)\cup B^\bullet.$ (3)

Then the events in B are said to be *concurrently a-enabled* under M_1.

Further we say that the marking M_2 is *reached* after the concurrent firing of the events in B. We call this firing an *a-step* and denote it by

$$M_1[B\rangle M_2 \tag{4}$$

We also say that B is a *step forward* from M_1 to M_2, and at the same time a *step backward* from M_2 to M_1.

It is called a *maximal a-step* if there is no B' with $B'\supset B$ (B' contains B properly) such that the events of B' can fire concurrently under M_1.

Definition 2.3. $\Sigma=(S,T,F,C)$ is called an *extended C/E system*–an *AC/E* system for short–if

(1) (S,T,F) is a net;

(2) C is a set of markings with the property that for any two different markings (called *cases* henceforth) M_1 and M_2, there exists a chain of forward and/or backward steps, which transforms M_1 into M_2; and

(3) For each $e\in T$, there exists a case $M\in C$, such that e is a-enabled under M.

Here the concepts "a-enabled" and "a-steps" are used because some pre-conditions of an event may coincide with some postconditions of the same event, if the net is nonpure. In that case it is difficult to decide whether the event has already fired.

Example 2.4. The event in Fig. 1 is a-enabled. Its (only) pre-condition coincides with its post-condition.

Corollary 2.5. *Any C/E system is an AC/E system.*

Definition 2.6. Let $\Sigma=(S,T,F,C)$ be an AC/E system. A triple $R=\{R[i]=(S_i,T_i,F_i)\mid i=1,2,3,\ldots\}$ is called an *NP/R net over* Σ (or a *NP/R net* for short) if the following conditions are satisfied.

(a) $K=(\bigcup S_i,\bigcup T_i,\bigcup F_i)$ is an occurrence net.

(b) $\forall i,\ F_i=\bigcup_{i\leqslant j}F_{ij}$,

$$\forall i\leqslant j,\quad F_{ij}\subseteq S_i\times T_j\cup T_i\times S_j. \tag{5}$$

(c) $\forall x\in S_i\cup T_i$, let

$$\begin{aligned}{}^{\bullet}x&=\{y\mid\exists j\leqslant i,\ (y,x)\in F_{ji}\},\\ x^{\bullet}&=\{y\mid\exists j\geqslant i,\ (x,y)\in F_{ij}\}.\end{aligned} \tag{6}$$

(d) If $S'=\bigcup S_i$, $T'=\bigcup T_i$, $F'=\bigcup F_i$, then there exists a mapping

$$p:S'\to S,\quad T'\to T,\quad F'\to F, \tag{7}$$

such that $\forall i$, $\forall x,y\in S_i\cup T_i$:

$$\begin{aligned}&(x,y)\in F'\to p((x,y))=(p(x),p(y))\in F,\\ &x\neq y\ \to\ p(x)\neq p(y),\end{aligned} \tag{8}$$

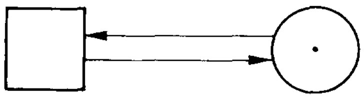

Fig. 1. An AC/E system.

and

$$\forall i,\ x \in T_i, \quad p({}^\bullet x) = {}^\bullet p(x) \wedge p(x^\bullet) = p(x)^\bullet. \tag{9}$$

We call $R[i]$ the ith *page* of R, and S_i, T_i, F_i the condition set, event set and arc set of $R[i]$, respectively. Finally, we call ${}^\bullet x$ and $x^\bullet$ the *preset* and *postset* of x, respectively.

Corollary 2.7. *It follows from* (8) *that for any* $x \in S_i \cup T_i$ *such that* $p(x) = y$ *we may use the notation* $y[i]$ $(=x)$.

Example 2.8. In Fig. 2, (a) and (b) are two AC/E systems, and (c) and (d) are two NP/R nets over them, where c has infinitely many pages.

Definition 2.9. Given an AC/E system Σ. The sequence $c_i[B_i\rangle c_{i+1}$, $1 \leqslant i \leqslant n$, is called a *step sequence* of Σ, where each element of the sequence is an a-step (a *step* for short). In the following, we will not differentiate between the terminologies step sequence, firing sequence, or a run. A successive subsequence of a step sequence is called a subrun. We denote the set of all event occurrences of the run A with $ev(A)$, and the ith occurrence of the event e in it with $e\langle i\rangle$.

Definition 2.10. Given a run A of an AC/E system Σ. Define the partial order of the event occurrences in $ev(A)$ as follows. Let A be composed of the step sequence $c_i[B_i\rangle c_{i+1}$, then:

(1) e_1 is before e_2, or in other words, e_2 is after e_1, if $i<k$, $e_1 \in B_i$, $e_2 \in B_k$.

(2) there is no other order among the event occurrences besides the one mentioned above.

Corollary 2.11. *The order given above is a partial order.*

Definition 2.12. Let R be an NP/R net. Define relations $\prec$, *rli* and *rco* as follows.

(a) If the elements $x[i]$, $y[j]$ of R, when considered as elements of the corresponding occurrence net (see Definition 2.6(a)), satisfy

$$x[i] < y[j],$$

then the following is also true:

$$x[i] \prec y[j]. \tag{10}$$

(b) $$\forall i<k,\ x[i] \prec y[k]. \tag{11}$$

(c) $$\forall i,j,k,\ x[i] \prec y[j] \ \wedge \ y[j] \prec z[k] \ \Rightarrow \ x[i] \prec z[k]. \tag{12}$$

(d) $$\forall i,j, \text{ if } x[i] \prec y[j], \text{ then one of (a)–(c) must be valid.} \tag{13}$$

(e) $$\forall i,j,\ x[i] = y[j] \ \vee \ x[i] \prec y[j] \ \vee \ y[j] \prec x[i] \ \Leftrightarrow \ x[i]\, rli\, y[j] \tag{14}$$

(f) $$\forall i,j,\ {\sim} x[i] \prec y[j] \ \wedge \ {\sim} y[j] \prec x[i] \ \Leftrightarrow \ x[i]\, rco\, y[j] \tag{15}$$

Theorem 2.13. (1) $x[i] \prec y[i] \Rightarrow x[i] < y[i]$,

(2) *relation* $\prec$ *is transitive,*

(3) *relation* $\prec$ *is anti-symmetric,*

(4) *therefore,* $\prec$ *is a partial order.*

Definition 2.14. Let R be a NP/R net, $Q \subseteq (\bigcup S_i) \cup (\bigcup T_j)$, then:

(1) Q is called an *r-line*, if $\forall x, y \in Q$, there is always $x\, rli\, y$, and $\forall z \notin Q$, $\exists t \in Q$, $\sim z\, rli\, t$;

(2) Q is called an *r-slice*, if $\forall x, y \in Q$, there is always $x\, rco\, y$, and $\forall z \notin Q$, $\exists t \in Q$, $\sim z\, rco\, t$.

Definition 2.15. Let $R = \{R[i] = (S_i, T_i, F_i) \mid i = 1, 2, 3, \dots\}$ be an NP/R net over the AC/E system $\Sigma = (S, T, F, C)$. Let A be a run of Σ. R is called an exact description of A, if the following conditions are satisfied:

(1) There is a bi-unique mapping β:

$$\beta : ev(A) \to \bigcup T_i. \tag{16}$$

(2) The mapping p of Definition 2.6 has the following properties:

(2.1) p is an internal mapping on each r-slice of R.

(2.2) Let α be a single-valued mapping which maps each event occurrence $a\langle i \rangle$ of $ev(A)$ to the event a of Σ, then we have

$$\forall t \in ev(A), \quad \alpha(t) = p(\beta(t)). \tag{17}$$

(3) For $t_1, t_2 \in ev(A)$, t_2 is behind t_1 if and only if $\beta(t_1) \prec \beta(t_2)$.

This mapping p is called a *process* of Σ.

Theorem 2.16. *Let* $\Sigma = (S, T, F, C)$ *be an* AC/E *system, then for each run* A *of* Σ, *there must be an NP/R net* R *over* Σ, *such that* R *is an exact description of* A.

Proof. Let A be a step sequence $B_1, B_2, B_3, \dots$. We construct the corresponding NP/R net, starting from B_1.

Step I. Assume that $\{e_1, \dots, e_l\}$ is the set of events firing concurrently in B_1. Let

$$^\bullet e_i = \{a_{i1}, \dots, a_{in_i}\}, \quad e_i^\bullet = \{b_{i1}, \dots, b_{im_i}\}, \qquad 1 \leqslant i \leqslant l. \tag{18}$$

Construct new objects $t_1, \dots, t_l$. For each i, construct new objects $x_{i1}, \dots, x_{in_i}$; $y_{i1}, \dots, y_{im_i}$. Let

$$X_i = \{x_{i1}, \dots, x_{in_i}\}, \quad Y_i = \{y_{i1}, \dots, y_{im_i}\}, \qquad 1 \leqslant i \leqslant l. \tag{19}$$

Define a mapping p:

$$\begin{aligned} &\forall i, j, \quad p(x_{ij}) = a_{ij}, \quad p(y_{ij}) = b_{ij}, \\ &p(t_i) = e_i, \qquad p((x, y)) = (p(x), p(y)). \end{aligned} \tag{20}$$

Call the t_i's, x_{ij}'s and y_{ij}'s separable objects, and their p-mappings e_i's, a_{ij}'s, b_{ij}'s original objects. Obviously, p is single-valued. Further let

$$F_{11}=\bigcup_{i=1}^{l}\bigcup_{j=1}^{n_i}\{(x_{ij},t_i)\},\qquad F_{12}=\bigcup_{i=1}^{l}\bigcup_{j=1}^{m_i}\{(t_i,y_{ij})\};\tag{21}$$

$$S_1=\bar{X}_1=\bigcup_{i=1}^{l}X_i,\quad \bar{Y}_1=\bigcup_{i=1}^{l}Y_i,\quad T_1=\bigcup_{i=1}^{l}\{t_i\},\quad F_1=F_{11}\cup F_{12};\tag{22}$$

$$R[1]=(S_1,T_1,F_1).$$

Step II. Assume that we have already constructed $R[1],\dots,R[k]$ for the steps $B_1,\dots,B_k$. Differentiate between two different cases.

Case 1: The number of steps of A is greater than k. Let

$$R[i]=(S_i,T_i,F_i),\quad 1\leqslant i\leqslant k.$$

At first, we construct a new page. Let

$$Y^{(k)}=\bigcup_{i=1}^{k}\bar{Y}_i,\qquad Y'=p(Y^{(k)}).\tag{23}$$

Note that above we have extended the mapping p, in such a way that it maps also sets of objects to sets of objects.

Furthermore, let the events that fire concurrently in B_{K+1} be $f_1,\dots,f_h$.

$$^{\bullet}f_i=(a_{i1},\dots,a_{in_i}),\quad f_i^{\bullet}=(b_{i1},\dots,b_{im_i}),\qquad 1\leqslant i\leqslant h.\tag{24}$$

Let

$$Y''=\left(\bigcup_{i=1}^{h}{}^{\bullet}f_i\right)\cap Y'.\tag{25}$$

If $Y''=\emptyset$, then construct $\bar{X}_{k+1}$, $\bar{Y}_{k+1}$, S_{k+1}, T_{k+1}, F_{k+1} and

$$R[k+1]=(S_{k+1},T_{k+1},F_{k+1})$$

in the same way as we constructed $R[1]$.

If $Y''\neq\emptyset$, let $Y'''=\{y\mid p(y)\in Y''\}$. Then

$$\forall i,\quad {}^{\bullet}f_i-Y''=\{a_{i1},\dots,a_{ig_i}\},\quad 0\leqslant g_i\leqslant n_i.\tag{26}$$

Generate new objects $u_1,\dots,u_h$. For each i, generate new objects $x_{i1},\dots,x_{ig_i}$; $y_{i1},\dots,y_{im_i}$. Let

$$X_i=\{x_{i1},\dots,x_{ig_i}\},\quad Y_i=\{y_{i1},\dots,y_{im_i}\},\qquad 1\leqslant i\leqslant h;\tag{27}$$

$$\begin{aligned}&\bar{X}_{k+1}=\bigcup_{i=1}^{h}X_i,\qquad \bar{Y}_{k+1}=\bigcup_{i=1}^{h}Y_i,\\&S_{k+1}=\bar{X}_{k+1}\cup\bar{Y}_k,\qquad T_{k+1}=\bigcup_{i=1}^{h}\{f_i\};\end{aligned}\tag{28}$$

$$F_{11}=\bigcup_{i=1}^{h}\bigcup_{j=1}^{g_i}\{(x_{ij},u_i)\},$$

$$F_{12}=\bigcup_{i=1}^{h}\{(y,u_i)\mid y\in\bar{Y}_k,\ p(y)\in{}^{\bullet}f_i\},\tag{29}$$

$$F_{13}=\bigcup_{i=1}^{h}\bigcup_{j=1}^{m_i}\{(u_i,y_{ij})\},$$

$$F_{k+1}=F_{11}\cup F_{12}\cup F_{13};$$

$$R[k+1]=(S_{k+1},\ T_{k+1},\ F_{k+1}).$$

Now, for $1\leqslant i\leqslant k$, modify the sets of objects constructed above as follows:

$$F'_i=\bigcup\{(x,y)\mid x\in\bar{Y}_{i-1}\cap Y''',\ y\in T_{k+1}\},$$

$$F_i:=F_i\cup F'_i,\tag{30}$$

$$\bar{Y}_i:=\bar{Y}_i-Y'''.\tag{31}$$

Note that := is the assignment symbol. It just replaces the content of the left-hand side by the right-hand side.

Extend the mapping p as follows:

$$\forall i,\ 1\leqslant i\leqslant h:\quad p(x_{ij})=a_{ij},\quad 1\leqslant j\leqslant g_i,$$

$$p(y_{ij})=b_{ij},\quad 1\leqslant j\leqslant m_i,\tag{32}$$

$$p(u_i)=f_i,$$

$$p((x,y))=(p(x),p(y)).$$

Case 2: The number of steps of A is k. Let

$$S_{k+1}=\bar{Y}_k,\qquad R[k+1]=(S_{k+1},\ \emptyset,\ \emptyset).\tag{33}$$

In this way, we get a (finitely or infinitely) multi-layered net on Σ.

$$R=\{R[i]=(S_i,T_i,F_i)\mid i=1,2,3,\ldots\}.$$

We prove that R is an NP/R net over Σ.

(i) In order to prove that $K_0=(S_0,T_0,F_0)$ is an occurrence net, where

$$S_0=\bigcup S_i,\qquad T_0=\bigcup T_i,\qquad F_0=\bigcup F_i,\tag{34}$$

we define the union of two occurrence nets as follows:

$$(K'=(S',T',F'))\cup(K''=(S'',T'',F''))$$

$$=(K'''=(S'\cup S'',T'\cup T'',F'\cup F''))\tag{35}$$

The proof proceeds inductively. It is easy to see that

$$\forall i,j,\quad |{}^{\bullet}x_{ij}|=|y_{ij}^{\bullet}|=0,\qquad |x_{ij}^{\bullet}|=|{}^{\bullet}y_{ij}|=1\tag{36}$$

was valid in the first step when constructing $R[1]$. Therefore $R[1]$ is an occurrence net (see e.g. [5, Section 3.2]).

Now assume that $R^{(k)}=R[1]\cup R[2]\cup\cdots\cup R[k]$ is an occurrence net. Our assertion is already proved if the run A has only k steps. Otherwise, construct $R[k+1]=(S_{k+1},T_{k+1},F_{k+1})$ by using the method given above. From the way we constructed $R[k+1]$ we know that $R[k+1]$ remains an occurrence net before the assignment formulas of (30). It has no common elements with $R^{(k)}$, i.e.

$$(S^{(k)}\cup T^{(k)})\cap(S_{k+1}\cup T_{k+1})=\emptyset. \tag{37}$$

In the assignment of (30) and thereafter, the modification to $R[k+1]$ is only the addition of some arcs from certain condition elements x (satisfying $x^{\bullet}=\emptyset$) of $R^{(k)}$ to certain event elements y (satisfying ${}^{\bullet}y=\emptyset$) of $R[k+1]$. Therefore, if we construct

$$R^{(k+1)}=R[k+1]\cup R^{(k)} \tag{38}$$

then all its condition elements x still satisfy $|{}^{\bullet}x|\leqslant 1$, $|x^{\bullet}|\leqslant 1$, and there is no cycle in it. That means $R^{(k+1)}$ is still an occurrence net.

(ii) The validity of (b)–(d) of Definition 2.6 is easy to prove.

(iii) In order to see the existence of the mapping p we need only to note that the mapping p produced when constructing R is precisely the mapping p required in Definition 2.6. The validity of requirements (8) and (9) can be checked in a straightforward way.

Thus we have shown that R is really a NP/R net over Σ. Now we prove that it is also an exact simulation of the run A (see Definition 2.15) as follows:

(i) From the way we constructed R we know, that there is an one-to-one correspondence between the objects of $\bigcup T_i$ and the event occurrences of $A=\{B_j\}$, where B_j are the steps of A. This correspondence is the mapping β required in Definition 2.15.

(ii) To prove that p is injective on each r-slice of R we assume that Q is such a r-slice, $x[i], y[j]\in Q$, $x[i]\neq y[j]$. Then $i=j$ follows from the definition of partial order of NP/R nets.

From (8) of Definition 2.16 we know that

$$p(x[i])\neq p(y[i]). \tag{39}$$

This shows that p is injective on Q.

(iii) To prove (17) we assume that $e\langle j\rangle$ (the jth occurrence of the event e in A) is in the ith step B_i of A. Then there is a unique element $t=\beta(e\langle j\rangle)$ corresponding to it, which is generated as a new object when constructing $R[i]$. It is $p(t)=e$ according to the definition of p. This is consistent with the fact $\alpha(e\langle j\rangle)=e$.

(iv) To prove property (3) of Definition 2.15, assume that $t_1,t_2\in ev(A)$ and t_2 is behind t_1. From Definition 2.10 we know $\exists B_i, B_j$, $t_1\in B_i$, $t_2\in B_j$, $i<j$. We know also that $\beta(t_1)$ and $\beta(t_2)$ are objects of $R[i]$ and $R[j]$ from the way we constructed R. Thus $\beta(t_1)\prec\beta(t_2)$ by (11) of Definition 2.12.

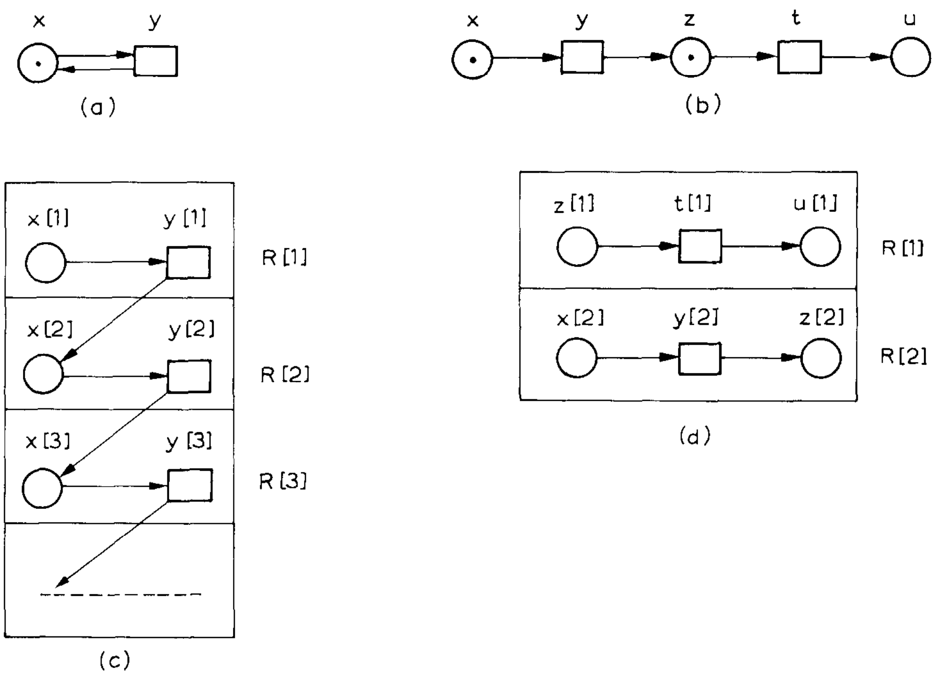

Fig. 2. Two NP/R nets.

The second part of the proof (validation of the other direction of the assertion) is similar to the first part and thus omitted here. □

Example 2.17. The NP/R net (c) given in Fig. 2 is an exact simulation of a run of the AC/E system in (a). Note that (c) was constructed using the algorithm given in Theorem 2.16. The NP/R net (d) of the same figure is an exact simulation of a run of the AC/E system in (b). But (d) was not constructed in the way (c) was.

Corollary 2.18. *Different NP/R nets may be exact simulations of the same run of the same AC/E system.*

3. The partial derivation semantics of CCS

In the following we reproduce briefly the results about CCS semantics given by Degano et al. based on AC/E systems.

$\Delta = \{\alpha, \beta, \gamma, \ldots\}$ is called the *action set* of CCS, $\bar{\Delta} = \{\bar{\alpha},\ \alpha \in \Delta\}$ its *anti-action set*. Actions of $\Delta \cup \bar{\Delta}$ are called *visible* ones; τ is the (only) invisible action, $\tau \notin \Delta \cup \bar{\Delta}$. The agents of CCS are defined with the following syntax:

$$e ::= x \mid nil \mid \alpha.e \mid e\backslash\alpha \mid e[s] \mid e_1 + e_2 \mid e_1|e_2 \mid \langle x\rangle.e \tag{40}$$

and the following derivation rules:

$$\alpha.e-\alpha\rightarrow e, \tag{41a}$$

$$e_1-\alpha\rightarrow e_2 \text{ implies } e_1\backslash\beta-\alpha-e_2\backslash\beta, \quad \text{where } \beta\notin\{\alpha,\bar{\alpha}\}, \tag{41b}$$

$$e_1-\alpha\rightarrow e_2 \text{ implies } e_1[s]-s(\alpha)\rightarrow e_2[s], \tag{41c}$$

$$e_1-\alpha\rightarrow e_2 \text{ implies } e_1+e-\alpha\rightarrow e_2 \text{ and } e+e_1-\alpha\rightarrow e_2, \tag{41d}$$

$$e_1-\alpha\rightarrow e_2 \text{ implies } e_1|e-\alpha-e_2|e \text{ and } e|e_1-\alpha-e|e_2, \tag{41e}$$

$$e_1-\alpha\rightarrow e_2 \text{ and } e'_1-\bar{\alpha}\rightarrow e'_2 \text{ imply } e_1|e'_1-\tau\rightarrow e_2|e'_2, \tag{41f}$$

$$e_1[\langle x\rangle.e_1/x]-\alpha\rightarrow e_2 \text{ implies } \langle x\rangle.e_1-\alpha\rightarrow e_2. \tag{41g}$$

The grapes are defined by Degano et al. [1] with the following syntax:

$$g ::= nil \mid \alpha.e \mid e+e \mid \langle x\rangle.e \mid id|e \mid e|id \mid e\backslash\alpha \mid e[s] \tag{42}$$

and the following decomposition rules:

$$dec(nil)=\{nil\}, \tag{43a}$$

$$dec(\alpha.e)=\{\alpha.e\}, \tag{43b}$$

$$dec(e\backslash\alpha)=dec(e)\backslash\alpha, \tag{43c}$$

$$dec(e[s])=dec(e)[s], \tag{43d}$$

$$dec(e_1+e_2)=\{e_1+e_2\}, \tag{43e}$$

$$dec(e_1|e_2)=dec(e_1)|id\cup id|dec(e_2), \tag{43f}$$

$$dec(\langle x\rangle.e)=\{\langle x\rangle.e\}. \tag{43g}$$

The partial order semantics of CCS is defined by the following derivation rules for grapes:

$$\{\alpha.e\}-\alpha\rightarrow dec(e), \tag{44a}$$

$$G_1-\alpha\rightarrow G_2 \text{ implies } G_1\backslash\beta-\alpha\rightarrow G_2\backslash\beta, \quad \beta\notin\{\alpha,\bar{\alpha}\}, \tag{44b}$$

$$G_1-\alpha\rightarrow G_2 \text{ implies } G_1[s]-s(\alpha)\rightarrow G_2[s], \tag{44c}$$

$$(dec(e_1)-G_3)-\alpha\rightarrow G_2 \text{ implies}$$
$$\{e_1+e\}-\alpha\rightarrow G_2\cup G_3 \text{ and } \{e+e_1\}-\alpha\rightarrow G_2\cup G_3, \tag{44d}$$

$$G_1-\alpha\rightarrow G_2 \text{ implies } G_1|id-\alpha\rightarrow G_2|id \text{ and } id|G_1-\alpha\rightarrow id|G_2, \tag{44e}$$

$$G_1-\alpha\rightarrow G_2 \text{ and } G_3-\bar{\alpha}\rightarrow G_4 \text{ imply}$$
$$G_1|id\cup id|G_3-\tau\rightarrow G_2|id\cup id|G_4, \tag{44f}$$

$$(dec(e_1[\langle x\rangle.e_1/x]-G_3)-\alpha\rightarrow G_2 \text{ implies } \{\langle x\rangle.e_1\}-\alpha\rightarrow G_2\cup G_3, \tag{44g}$$

where G_1, G_2, G_3, G_4 are sets of grapes. A set G of grapes is called complete, if there exists an agent e such that $dec(e) = G$. It was proved [1] that the function *dec* defines an one-to-one correspondence between agents and complete sets of grapes. It was also proved that if $G - \alpha \rightarrow G_2$ is a derivation of grapes, then there must be a set G_3 of grapes, such that $G_1 \cap G_3 = \emptyset$ and $G_1 \cup G_3$ is a complete set of grapes. Furthermore, we must have $G_2 \cap G_3 = \emptyset$ and $G_2 \cup G_3$ is also a complete set of grapes.
The system $\Sigma_{\mathrm{CCS}} = (S, T, F, C)$ is defined as follows:

- $S =$ the set of all grapes,
- $T =$ the set of all derivations of grapes,
- $F = \{(g_1, G_1 - \alpha \rightarrow G_2), (G_1 - \alpha \rightarrow G_2, g_2) \mid g_1 \in G_1,\ g_2 \in G_2,\ G_1 - \alpha \rightarrow G_2 \in T\}$,
- $C =$ the set of all complete sets of grapes.

It was proved that Σ_{CCS} is an AC/E system.

4. A true concurrency semantics for CCS

In order to solve the problems which have not been solved in [1], we extend the syntax of CCS a little bit and then give its operational semantics in the form of derivation rules.

Definition 4.1. The CCS syntax is extended as follows:

$$e ::= x \mid nil \mid \alpha.e \mid e \backslash \alpha(a) \mid e[s] \mid e_1 + e_2 \mid e_1 | e_2 \mid e_1 \text{ where } x = e_2, \tag{45}$$

where a is an arbitrary identifier.

Let H_i, H_i', H_i'', $i = 0, 1, \ldots$ be multisets (i.e. bags) with the elements of $\Delta \cup \bar{\Delta} \cup \{\tau\}$ as their elements. $V(H_i)$ is the multiset of all visible actions in H_i, whereas $inv(H_i)$ is the multiset of all invisible actions (τ) of H_i. Then we have the following derivation rules

$$\alpha.e - \{\alpha\} \rightarrow e, \tag{46a}$$

$e_1 - H_0 \rightarrow e_2$ implies

$$e_1 \backslash \alpha(a) - H_0 \rightarrow e_2 \backslash \alpha(a), \quad \alpha \notin H_0 \cup \bar{H}_0$$

$$e_1 \backslash \alpha(a) - (H_0 - \{\alpha, \bar{\alpha}\}) \rightarrow e_2 \backslash \alpha(a), \quad \text{for all } \alpha.$$

where $H_0 - \{\alpha, \bar{\alpha}\}$ means "delete all occurrences of α and $\bar{\alpha}$ from H_0", (46b)

$$e_1 - H_0 \rightarrow e_2 \text{ implies } e_1[s] - s(H_0) \rightarrow e_2[s], \tag{46c}$$

$$e_1 - H_0 \rightarrow e_2 \text{ implies } e_1 + e - H_0 \rightarrow e_2 \text{ and } e + e_1 - H_0 \rightarrow e_2, \tag{46d}$$

$$e_1 - H_0 \rightarrow e_2 \text{ implies } e_1 | e - H_0 \rightarrow e_2 | e \text{ and } e | e_1 - H_0 \rightarrow e | e_2, \tag{46e}$$

$$e_1 - H_1 \to e_2 \text{ and } e_3 - H_2 \to e_4, \quad \alpha \in \{\beta, \bar{\beta}\} \quad \text{imply}$$

$$e_1 | e_3 - H_4 \to e_2 | e_4 \text{ and } e_1 \backslash \alpha(a) | e_3 \backslash \beta(a) - H'_4 \to e_2 \backslash \alpha(a) | e_4 \backslash \beta(a), \tag{46f}$$

$$e_1[(e_2 \text{ where } x = e_2)/x] - H_0 \to e_3 \text{ where } x = e_2$$

$$\text{implies } (e_1 \text{ where } x = e_2) - H_0 \to e_3 \text{ where } x = e_2. \tag{46g}$$

Here we have used the following notations:

$$H_4 = H'_1 \cup H'_2 \cup inv(H_1) \cup inv(H_2) \cup inv(|H_3|),$$

$$H'_4 = H''_1 \cup H''_2 \cup inv(H_1) \cup inv(H_2) \cup inv(|H_3|),$$

$$H'_1 = v(H_1) - H_3, \qquad H'_2 = v(H_2) - \bar{H}_3,$$

$$H''_1 = H'_1 - \{\alpha, \bar{\alpha}\}, \qquad H''_2 = H'_2 - \{\alpha, \bar{\alpha}\},$$

$$H_3 \subseteq v(H_1) \cap \overline{v(H_2)},$$

where the intersection of two multisets is defined as the maximal sub-multiset which is contained in both multisets. $|H_i|$ is the number of elements contained in multiset H_i, $inv(|H_i|)$ is the multiset of $|H_i|$ τ's. $\bar{H}_i = \{\bar{\alpha} \mid \alpha \in H_i\}$.

Note that this concurrency semantics of CCS is different from both Milner's and that of Degano et al. It will contain the Milner semantics as its proper subset by adding the following two rules.

(a) Let e_1 where $x = e_1$ be equal to $\langle x \rangle . e_1$ in the original syntax.

(b) Let the agents in which the identifiers "a" in operations $\backslash \alpha(a)$ are all different from each other equal to those agents in the original syntax, which have the same form except for the identifiers which do not exist.

As to the derivation rules, note that the rules (46b)–(46e) and (46g) will be the same as those in Milner's semantics if $|H_0| = 1$. The derivation (46f) will be the same as that in Milner's semantics if $|H_1| = |H_2| = 1$, $v(H_1) = \overline{v(H_2)} \neq \emptyset$. Rule (46a) as it stands is also valid for Milner's semantics.

Example 4.2. Let

$$e_1 = (\alpha.nil | \beta.nil) | ((\bar{\alpha}.nil | \bar{\gamma}.nil) | (\omega.nil | \varphi.nil)),$$

$$e_2 = ((\delta.nil | \bar{\beta}.nil) | (\gamma.nil | \bar{\delta}.nil)) | (\bar{\omega}.nil | \psi.nil).$$

It follows from (46f) that

$$e_1 - \{\tau, \beta, \bar{\gamma}\} \to (nil | nil) | ((nil | nil) | (\omega.nil | \varphi.nil)),$$

$$e_2 - \{\tau, \bar{\beta}, \gamma\} \to ((nil | nil) | (nil | nil)) | (\bar{\omega}.nil | \psi.nil)$$

implies

$$e_1|e_2-\{\tau,\tau,\beta,\bar{\beta},\tau\}\rightarrow((nil|nil)|((nil|nil)|(\omega.nil|\varphi.nil)))$$
$$|(((nil|nil)|(nil|nil))|(\bar{\omega}.nil|\psi.nil))$$

where

$$H_3=\{\bar{\gamma}\}\subseteq(\{\beta,\bar{\gamma}\}\cap\{\beta,\bar{\gamma}\})=\{\beta,\bar{\gamma}\},$$
$$H'_1=\{\beta\},\qquad H'_2=\{\bar{\beta}\},$$
$$inv(H_1)=inv(H_2)=inv(|H_3|)=\{\tau\},$$
$$H_4=\{\tau,\tau,\beta,\bar{\beta},\tau\}.$$

Definition 4.3. A relation $SB\subseteq E\times E$, where E is the set of all agents, is called a *bisimulation* if the following conditions are fulfilled:

(a) if $(e_1,e_2)\in SB$, $e_1-H\rightarrow e'_1$, then there exists e'_2, such that $e_2-H\rightarrow e'_2$, and $(e'_1,e'_2)\in SB$,

(b) $(e_1,e_2)\in SB\Leftrightarrow(e_2,e_1)\in SB$ where H is a multiset of actions.

Theorem 4.4. *For each agent e there exists another agent $N(e)$ such that*

$$N(e)=\sum_{i=1}^{m}\prod_{j=1}^{n_i}e_{ij}\quad\textit{and}\quad(N(e),e)\in SB,\tag{49}$$

where

$$\sum_{i=1}^{n}e_i=e_1+e_2+\cdots+e_n,$$
$$\prod_{i=1}^{n}e_i=[e_1|e_2|e_3|\cdots|e_n]$$

(*the n-ary product $\prod_{i=1}^{n}e_i$ is appropriately composed of binary products $(e'|e'')$ by using parentheses. Each e_{ij} takes one of the following forms*:

$$\sum_{i=1}^{k}\alpha_i.e'_i\backslash A_i\text{ where }x_i=e''_i\quad\textit{or}\quad nil,$$

where each α_i is a (*visible or invisible*) *action. e'_i and e''_i take the form* (49). *A_i is a set of actions with identifier parameters $\{\alpha_1(a_1),\ldots,\alpha_l(a_l)\}$, $\forall i\neq j$, $\alpha_i\neq\alpha_j$. A_i may be empty. To avoid details, the naming action $e[s]$ is not considered here.*

Proof. We use the following transformation rules to prove this theorem:

$$(e_{11}+e_{12})|(e_{21}+e_{22})=e_{11}|e_{21}+e_{11}|e_{22}+e_{12}|e_{21}+e_{12}|e_{22},\tag{50a}$$

$$\left(\sum_{i=1}^{n}e'_i\right)\text{where }x=e''\;=\;\sum_{i=1}^{n}(e'_i\text{where }x=e''),\tag{50b}$$

$$\left(\prod_{i=1}^{n} e_i'\right) \text{where } x_i'=e_i'' \;=\; \prod_{i=1}^{n} (e_i' \text{where } x_i'=e_i''), \tag{50c}$$

$$e_i' \text{where } x_i'=e_i'' \;=\; e_i' \quad \text{where } e_i' \text{ does not contain } x_i', \tag{50d}$$

$$(e_i' \text{where } x_i'=e_i'')\backslash A_i \;=\; (e_i'\backslash A_i) \text{where } x_i'=e_i'', \tag{50e}$$

$$e_i\backslash A_i\backslash A_i'=e_i\backslash(A_i\cup A_i''), \quad \text{where}$$

$$A_i''=\{\alpha(a)\,|\,\alpha(a)\in A_i', \sim\exists b\neq a, \text{ such that } \alpha(b)\in A_i \text{ or } \bar{\alpha}(b)\in A_i\}, \tag{50f}$$

$$(\textstyle\sum e_i)\backslash A=\textstyle\sum(e_i\backslash A), \tag{50g}$$

$$(\textstyle\prod e_i)\backslash A=\textstyle\prod(e_i\backslash\{\alpha(c)\,|\,\alpha(a)\in A\})$$

where c is a completely new identifier. (50h)

$$nil\backslash A=nil. \tag{50i}$$

To avoid going too far into the details, we assume here that each recursive agent has only one agent variable. Different recursive agents use different identifiers to denote their agent variables. We assume also that there is no nested recursive agent.

Using induction it is easy to prove that each agent can be transformed into the form (49) by using the transformation rules given above.

Now we will prove that each of these transformations keeps the bisimulation relation unchanged.

Transformation (50a). Let

$$e=(e_{11}+e_{12})|(e_{21}+e_{22}),$$

$$f=e_{11}|e_{21}+e_{11}|e_{22}+e_{12}|e_{21}+e_{12}|e_{22}.$$

Given a derivation $e-H\rightarrow e'$, let (see Definition 4.1):

$$H=H_1\cup H_2, \qquad H_1=H_{11}\cup H_{12},$$

$$H_2=H_3\cup H_4, \qquad H_4=H_{41}\cup H_{42};$$

$$e_1=e_{11}+e_{12}, \qquad e_2=e_{21}+e_{22}$$

where H_{11} and H_{12} are the sets of visible actions executed by e_1 and e_2, respectively. H_{41} and H_{42} are the sets of invisible actions executed by e_1 and e_2, respectively. H_3 is the set of invisible actions produced during the communication between e_1 and e_2. All the sets mentioned above are multisets.

W.l.o.g., assume that H_{11} and H_{12} are the sets of visible actions produced by e_{11} and e_{21} respectively. H_{41} and H_{42} are the sets of invisible actions produced by e_{11} and e_{21} respectively. H_3 is the set of invisible actions produced during the communication between e_{11} and e_{21}. That means

$$e_{11}|e_{21}-H\rightarrow e_{11}'|e_{21}'' \;(=e').$$

From (46d) follows the existence of derivation $f-H\rightarrow e'$. On the other hand, if any derivation $f-H\rightarrow f'$ is given, we can also prove the existence of the corresponding derivation $e-H\rightarrow f'$ in the same way. That means $(e,f)\in SB$.

Transformation (50b). Let

$$e=\left(\sum_{i=1}^{n} e'_i\right)\text{where } x=e'', \qquad f=\sum_{i=1}^{n}(e'_i \text{ where } x=e'').$$

Consider a derivation

$$e-H\rightarrow e' \text{ where } x=e''. \tag{51}$$

It follows from (46g), that derivation (51) can only occur when there is another derivation of the following form:

$$\left(\sum_{i=1}^{n} e'_i\right)[(e'' \text{ where } x=e'')/x]-H\rightarrow e' \text{ where } x=e''. \tag{52}$$

This can be rewritten as follows:

$$\sum_{i=1}^{n} e'_i[(e'' \text{ where } x=e'')/x]-H\rightarrow e' \text{ where } x=e''.$$

Therefore there exists a j, such that

$$e'_j[(e'' \text{ where } x=e'')/x]-H\rightarrow e' \text{ where } x=e''.$$

It follows again from (46g) that

$$(e'_j \text{ where } x=e'')-H\rightarrow e' \text{ where } x=e''.$$

Thus we have

$$\sum_{i=1}^{n}(e'_i \text{ where } x=e'')-H\rightarrow e' \text{ where } x=e''.$$

On the other hand, each derivation $f-H\rightarrow f'$ where $x=f''$ implies also the existence of a derivation $e-H\rightarrow f'$ where $x=f''$. This means $(e,f)\in SB$.

Transformation (50c). The proof is similar to that of Transformation (50b).

Transformation (50d). Proof is obvious.

Transformation (50e). Proof is obvious.

Transformation (50f). Let

$$e=e_1\backslash A_1\backslash A_2, \tag{53}$$

$$f=e_1\backslash A_1\cup A_3 \tag{54}$$

where

$$A_3=\{\alpha(a)\mid \alpha(a)\in A_2, \sim\exists b\neq a, \text{ such that } \alpha(b)\in A_1 \text{ or } \bar{\alpha}(b)\in A_1\}.$$

Consider the derivation

$$e - H \rightarrow e' \backslash A_1 \backslash A_2. \tag{55}$$

It follows from (46f) that the following derivation is valid:

$$e_1 - H \rightarrow e', \quad \forall \alpha(a) \in A_1 \cup A_2, \ \alpha \notin H \cup \bar{H}.$$

Since $(A_1 \cup A_3) \subseteq (A_1 \cup A_2)$ we know that $\forall \alpha(a) \in A_1 \cup A_3, \alpha \notin H \cup \bar{H}$. Thus the validity of the following derivation is assured by (46f):

$$f - H \rightarrow e' \backslash (A_1 \cup A_3). \tag{56}$$

Note that the forms of (55) and (56) are similar to those of (53) and (54), respectively. Therefore we can repeat the reasoning described above and prove that for all n,

$$e^{(n)} - H \rightarrow e_1^{(n)} \backslash A_1 \backslash A_2$$

implies

$$f^{(n)} - H \rightarrow e_1^{(n)} \backslash (A_1 \cup A_3)$$

where $e^{(n)}, f^{(n)}, e_1^{(n)}$ are the nth derivations of e, f and e_1 respectively.

On the other hand, any

$$f^{(n)} - H \rightarrow e_1^{(n)} \backslash (A_1 \cup A_3)$$

implies

$$e^{(n)} - H \rightarrow e_1^{(n)} \backslash A_1 \backslash A_2$$

because $\{\alpha \mid \exists a, \ \alpha(a) \in A_2 \text{ or } \bar{\alpha}(a) \in A_2\} = \{\alpha \mid \exists a, \ \alpha(a) \in A_3 \text{ or } \bar{\alpha}(a) \in A_3\}$. Hence $(e, f) \in SB$.

Transformation (50g). Proof is obvious.

Transformation (50h). To shorten the proof, we consider only the cases of binary products and $|A| = 1$. Let

$$e = (e_1 | e_3) \backslash \alpha(a), \tag{57}$$

$$f = (e_1 \backslash \alpha(c)) | (e_3 \backslash \alpha(c)) \tag{58}$$

where c is a new identifier. Assume

$$e - H_4 \rightarrow e' \backslash \alpha(a) \tag{59}$$

then the following must hold:

$$e_1 | e_3 - H_4 \rightarrow e', \quad \alpha \notin H_4 \cup \bar{H}_4.$$

This is only possible when

$$e_1 - H_1 \rightarrow e_2, \quad e_3 - H_2 \rightarrow e_4, \qquad e_2 | e_4 = e'.$$

It then follows that

$$(e_1 \backslash \alpha(a)) | (e_3 \backslash \alpha(a)) - H_4' \rightarrow (e_2 \backslash \alpha(a)) | (e_4 \backslash \alpha(a)).$$

For the meaning of H_1, H_2, H_4 see (46f). Note that the only difference between H_4 and H'_4 is that H_4 may have α or $\bar{\alpha}$ which do not exist in H'_4. But in our case we have $\alpha \notin H_4 \cup \bar{H}_4$, therefore the derivation above can be rewritten as follows:

$$(e_1 \backslash \alpha(a)) | (e_3 \backslash \alpha(a)) - H_4 \rightarrow (e_2 \backslash \alpha(a)) | (e_4 \backslash \alpha(a))$$

This is valid and does not depend on the choice of a. Therefore we have

$$f - H_4 \rightarrow f', \quad f' = (e_2 \backslash \alpha(c)) | (e_4 \backslash \alpha(c)). \tag{60}$$

By comparing (57), (58) with (59), (60) we know that the same way of reasoning can be applied to e' and f', and $e^{(n)}$, $f^{(n)}$ for arbitrary n. On the other hand, if

$$f - H'_4 \rightarrow f' \quad (= (e_2 \backslash \alpha(c)) | (e_4 \backslash \alpha(c)))$$

is given, then it follows from (46f) that

$$e_1 - H_1 \rightarrow e_2, \qquad e_3 - H_2 \rightarrow e_4, \qquad e_1 | e_3 - H_4 \rightarrow e_2 | e_4.$$

This implies:

$$(e_1 | e_3) \backslash \alpha(a) - (H_4 - \{\alpha, \bar{\alpha}\}) \rightarrow (e_2 | e_4) \backslash \alpha(a),$$

$$(e_1 | e_3) \backslash \alpha(a) - H'_4 \rightarrow (e_2 | e_4) \backslash \alpha(a).$$

This way of reasoning can be applied to any $e^{(n)}$ and $f^{(n)}$ for arbitrary n, thus $(e, f) \in SB$. □

Discussion. The key point of transforming any agent e into $N(e)$ (sometimes we call it the normal form of e, although $N(e)$ is not uniquely determined by e) is not to lose the global information of the agent, especially for recursive agents "e where $x = e$" and masked agents "$e \backslash A$". See (b), (c) and (h) of the following example.

Example 4.5.

(a) $(e_1 | e_2 + e_3 | e_4) | (e_5 | e_6 + e_7 | e_8) = (e_1 | e_2) | (e_5 | e_6) + (e_1 | e_2) | (e_7 | e_8)$
$\qquad + (e_3 | e_4) | (e_5 | e_6) + (e_3 | e_4)(e_7 | e_8)$.

(b) $(\alpha.x + \beta.x \text{ where } x = \alpha.x + \beta.x) = (\alpha.x \text{ where } x = \alpha.x + \beta.x)$
$\qquad + (\beta.x \text{ where } x = \alpha.x + \beta.x)$

(c) $(\alpha.x | \beta.x \text{ where } x = \alpha.x + \beta.x)$
$\qquad = (\alpha.x \text{ where } x = \alpha.x + \beta.x) | (\beta.x \text{ where } x = \alpha.x + \beta.x)$

(d) $(\alpha.\beta.nil \text{ where } x = \gamma.x) = \alpha.\beta.nil$

(e) $(\alpha.x \text{ where } x = \beta.x) \backslash \{\beta(a)\} = (\alpha.x \backslash \beta(a)) \text{ where } x = \beta.x$

(f) $e \backslash \{\alpha(a), \beta(a)\} \backslash \{\alpha(b), \gamma(b)\} = e \backslash \{\alpha(a), \beta(a), \gamma(b)\}$

(g) $(\alpha.nil+\beta.nil)\backslash\{\alpha(b)\}=\alpha.nil\backslash\{\alpha(b)\}+\beta.nil\backslash\{\alpha(b)\}$

(h) $((\alpha.nil|\bar{\alpha}.nil)\backslash\{\alpha(b)\}|\alpha.nil)\backslash\{\alpha(a)\}$
$=(\alpha.nil\backslash\{\alpha(c)\})|(\bar{\alpha}.nil\backslash\{\alpha(c)\})|(\alpha.nil\backslash\{\alpha(a)\})$

Definition 4.6. All agents in normal form (or their grapes) can be decomposed into sets of grapes (or their sub-grapes) in the following way:

$$dec(nil)=\{nil\}, \tag{61a}$$

$$dec(\alpha.e)=\{\alpha.e\}, \tag{61b}$$

$$dec(e\backslash\alpha(a))=dec(e)\backslash\alpha(a), \tag{61c}$$

$$dec(e[s])=dec(e)[s], \tag{61d}$$

$$dec(\textstyle\sum e_i)=\textstyle\sum dec(e_i), \tag{61e}$$

$$dec\left(\prod_{i=1}^{n} e_i\right)=\left\{\prod_{i=1}^{j} id|dec(e_{j+1})|\prod_{i=j+2}^{n} id \,\middle|\, j=0,\ldots,n-1\right\}, \tag{61f}$$

$$dec\left(\prod_{i=l}^{m} id|e|\prod_{j=k}^{m} id\right)=\left\{\prod_{i=l}^{m} id|dec(e)|\prod_{j=k}^{m} id\right\}, \tag{61g}$$

where the products of *id* do not exist if $l>m$ or $k>m$ (but always $\min(l,k)\leqslant m$).

$$dec(e_1 \text{ where } x=e_2)=\{dec(e_1) \text{ where } x=e_2\}. \tag{61h}$$

Note that e, e_1 and e_2 can be any agents. In rule (61h), e_1 can be an agent or a grape. Here we allow the operands of the sum operator "+" and the product operator "|" to be sets of grapes, i.e.

$$\sum_{i=1}^{n} G_i=\left\{\sum_{i=1}^{n} g_i \,\middle|\, g_i\in G_i\right\} \quad \text{and} \quad \prod_{i=1}^{n} G_i=\left\{\prod_{i=1}^{n} g_i \,\middle|\, g_i\in G_i\right\}.$$

Remember that each product of agents and/or grapes must be combined into binary products appropriately. Further we define:

$$\{G \text{ where } x=e\}=\{g \text{ where } x=e \mid g\in G\}.$$

Definition 4.7. The *concurrent derivation rules* of grapes are the following:

$$\{\alpha.e\}-\alpha\rightarrow dec(e), \tag{62a}$$

$$G_1-\alpha\rightarrow G_2 \text{ implies } G_1\backslash\beta(a)-\alpha\rightarrow G_2\backslash\beta(a), \quad \beta\notin\{\alpha,\bar{\alpha}\}, \tag{62b}$$

$$G_1-\alpha\rightarrow G_2 \text{ implies } G_1[s]-s(\alpha)\rightarrow G_2[s] \tag{62c}$$

$$(dec(e_1)-G_1)-\alpha\rightarrow G_2 \text{ implies } (dec(e_1)-G_1)+dec(e_2)-\alpha\rightarrow G_2 \text{ and}$$
$$dec(e_2)+(dec(e_1)-G_1)-\alpha\rightarrow G_2, \tag{62d}$$

$G_1 - \alpha \rightarrow G_2$ implies $G_1 | id - \alpha \rightarrow G_2 | id$ and

$$id | G_1 - \alpha \rightarrow id | G_2, \tag{62e}$$

$G_1 - \alpha \rightarrow G_2$ and $G_3 - \bar{\alpha} \rightarrow G_4$ imply

$$G_1 | id \cup id | G_3 - \tau \rightarrow G_2 | id \cup id | G_4$$
$$G_1 \backslash \alpha(a) | id \cup id | G_3 \backslash \beta(a) - \tau$$
$$\rightarrow G_2 \backslash \alpha(a) | id \cup id | G_4 \backslash \beta(a), \quad \alpha \in \{\beta, \bar{\beta}\}, \tag{62f}$$

$G_1 - \alpha \rightarrow G_2$ implies

$$G_1 \text{ where } x = e - \alpha \rightarrow dec(G_2[e/x]) \text{ where } x = e, \tag{62g}$$

provided that G_1 does not contain something like "where $x = e_1$".

Example 4.8.

$$dec(e_1 | e_2 + e_3 | e_4) = dec(e_1 | e_2) + dec(e_3 | e_4)$$
$$= \{e_1 | id, id | e_2\} + \{e_3 | id, id | e_4\}$$
$$= \{e_1 | id + e_3 | id, e_1 | id + + id | e_4, id | e_2 + e_3 | id, id | e_2 + id | e_4\}$$

provided that $\forall i$, $dec(e_i) = e_i$.

Example 4.9.

$$dec((\alpha.x | \beta.x + \gamma.x | \delta.x) \text{ where } x = \alpha.x | \beta.x + \gamma.x | \delta.x$$
$$= dec(\alpha.x | \beta.x + \gamma.x | \delta.x) \text{ where } x = \alpha.x | \beta.x + \gamma.x | \delta.x$$
$$= \{\alpha.x | id + \gamma.x | id, \alpha.x | id + id | \delta.x, id | \beta.x + \gamma.x | id,$$
$$id | \beta.x + id | \delta.x\} \text{ where } x = \alpha.x | \beta.x + \gamma.x | \delta.x$$
$$= \{\alpha.x | id + \gamma.x | id \text{ where } x = \alpha.x | \beta.x + \gamma.x | \delta.x,$$
$$\alpha.x | id + id | \delta.x \text{ where } x = \alpha.x | \beta.x + \gamma.x | \delta.x,$$
$$id | \beta.x + \gamma.x | id \text{ where } x = \alpha.x | \beta.x + \gamma.x | \delta.x,$$
$$id | \beta.x + id | \delta.x \text{ where } x = \alpha.x | \beta.x + \gamma.x | \delta.x\}.$$

Example 4.10. It follows from rules (62d), (62e) and Example 4.8 that $e_1 - \alpha \rightarrow e'_1$ implies

$$\{e_1 | id + e_3 | id, e_1 | id + id | e_4\} - \alpha \rightarrow e'_1 | id.$$

Example 4.11. It follows from rules (62d), (62e), (62g) and Example 4.9 that:

$$\{\alpha.x | id + \gamma.x | id \text{ where } x = \alpha.x | \beta.x + \gamma.x | \delta.x,$$
$$\alpha.x | id + id | \delta.x \text{ where } x = \alpha.x | \beta.x + \gamma.x | \delta.x\} - \alpha$$
$$\rightarrow dec(\alpha.x | \beta.x + \gamma.x | \delta.x) | id \text{ where } x = \alpha.x | \beta.x + \gamma.x | \delta.x.$$

The representation of a CCS agent by an AC/E system is not perfect. In fact, the information contained in it is not complete, there is no initial case (marking) in an AC/E system. In the following we give a more exact way of representing an agent.

Definition 4.12. If $\Sigma=(S,T,F,C)$ is an AC/E system, then $\Sigma_1=(S,T,F,C,M)$ is called an *AC/EN system*, where $M\in C$ is called the *initial marking* of Σ_1. The firing rules of AC/EN systems are the same as those of AC/E systems.

Definition 4.13. An AC/EN system (S,T,F,C,M) is called *reachable*, if any case of C is reachable by foward firing, starting from the initial marking.

Definition 4.14. A set G of grapes is called a *direct unfolding* of an agent e, if $G=dec(e)$. G is called a *unfolding* of e if

- G is a direct unfolding of e, or
- there exists a unfolding G' of e and a derivation:

$$(G'-G'')-\alpha\rightarrow(G-G'')$$

where G'' is a set of grapes, α is an action of CCS.

A grape g is called a *direct descendent* of an agent e, if g is an element of $dec(e)$. g is called a descendent of e, if g is an element of some unfolding of e.

Definition 4.15. The *net-representation* of an agent e is an AC/EN system $\Sigma(e)=(S,T,F,C,M)$, constructed as follows:

$$M=dec(e), \tag{63a}$$

$$C=\{a\mid a \text{ is an unfolding of } e\}, \tag{63b}$$

$$S=\{b\mid b \text{ is a descendent of } e\}, \tag{63c}$$

$$T=\{(G'-G'')-H\rightarrow(G-G'')\mid G' \text{ and } G \text{ are both unfoldings of } e\}, \tag{63d}$$

$$F=\{(x,y)\mid x\in(G'-G'')\}\cup\{(y,z)\mid z\in(G-G'')\}, \tag{63e}$$

where $y\in T$, and y takes the form $(G'-G'')-H\rightarrow(G-G'')$.

It is easy to prove the following theorem.

Theorem 4.16. *$\Sigma(e)=(S,T,F,C,M)$ as it is constructed in Definition* 4.15 *is really an AC/EN system.*

Example 4.17. Reconsider Example 4.8 with $e_1=\alpha.nil$, $e_2=\beta.nil$, $e_3=\gamma.nil$, $e_4=\delta.nil$. Let a,b,c,d denote the four grapes in $dec(e_1|e_2+e_3|e_4)$ in the order as it was given there. Then $\Sigma(e)$ is shown in Fig. 3, where the simplified notations $\alpha,\beta,\gamma,\delta$ denote the four events $\{a,b\}-\alpha\rightarrow\{nil|id\}$, etc.

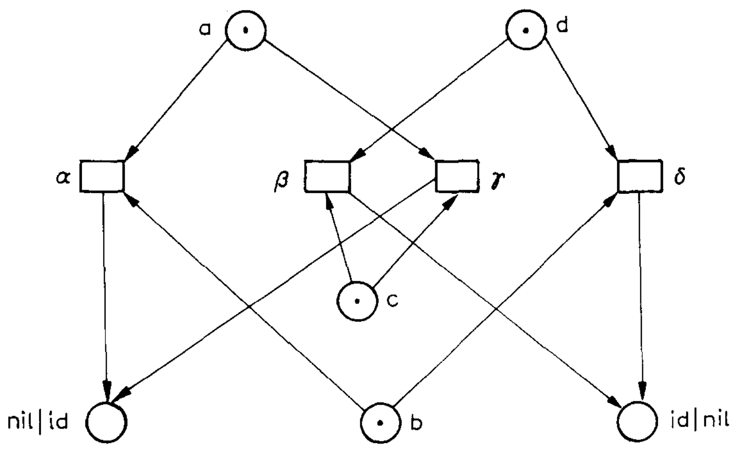

Fig. 3.

Example 4.18. Fig. 3 can also be understood as the net-representation of the agent e in Example 4.9, if we give a new explanation to it. Namely, let $e = \alpha.x | \beta.x + \gamma.x | \delta.x$. Let a, b, c, d denote the four grapes of $dec(e)$. Let $\alpha, \beta, \gamma, \delta$ denote the four grape derivations (one of which was shown in Example 4.11). Rename the elements $nil | id$ and $id | nil$ in Fig. 3 as $e | id$ where $x = e$ and $id | e$ where $x = e$. Then Fig. 3 contains all possible unfoldings of Example 4.9 for the "first step".

5. Bisimulation and semantical equivalence

Definition 5.1. Let R and R' be NP/R nets. Their occurrence net representations (see Definition 2.6(a)) are:

$$K = (S, T, F), \qquad K' = (S', T', F').$$

We say that R is *t-isomorphic* to R', if there is a mapping

$$\Phi : T \to T'$$

such that

$$\Phi \text{ is a bi-unique mapping,} \tag{64a}$$

$$\Phi(a[i]) = b[j] \;\to\; \forall a[k], \exists m, \Phi(a[k]) = b[m], \tag{64b}$$

$$a[i] \prec b[j] \;\leftrightarrow\; \Phi(a[i]) \prec \Phi(b[j]). \tag{64c}$$

Condition (64b) guarantees that if two elements x, y of a NP/R net R correspond to the same element of the underlying AC/EN system, and if R is t-isomorphic to R', then the mappings x', y' of x, y in R' correspond also to the same event of its underlying AC/EN system. Condition (64c) keeps the partial order unchanged.

Definition 5.2. Let e_1 and e_2 be two agents. R_1 and R_2 are NP/R nets over $\Sigma(e_1)$ and $\Sigma(e_2)$, respectively. We say that R_1 is *b-isomorphic* to R_2, if we change (64b) as follows:

$$\Phi \text{ maps the event } G_1 - \alpha \rightarrow G_2 \text{ of } R_1 \text{ to an event } G_3 - \alpha \rightarrow G_4 \text{ of } R_2. \tag{64b$'$}$$

Definition 5.3. Let $R = \{R[i] = (S_i, T_i, F_i) \mid i = 1, 2, \ldots\}$ be a NP/R net, then

$$R[i] \oplus n = (S, T, F) \tag{65}$$

where

$$S = \{x[i+n] \mid x[i] \in S_i\}, \qquad T = \{x[i+n] \mid x[i] \in T_i\},$$

$$F = \{(x[i+n], y[j+n]) \mid (x[i], y[j]) \in F_i\},$$

$$R \oplus n = \{R[i] \oplus n \mid i = 1, 2, \ldots\}.$$

Definition 5.4. Let $R = \{R[i] \mid i = 1, 2, \ldots, n\}$ and $R' = \{R'[i] \mid i = 1, 2, 3, \ldots\}$ be two NP/R nets over the same AC/EN system Σ. We define the concatenation R'' of R and R' as follows.

Let

$$\forall i \leqslant n, \quad R''[i] = R[i],$$

$$\forall i > n, \quad R''[i] = R'[i-n] \oplus n. \tag{66}$$

Modify R'' as follows. If

$$K = (S, T, F), \qquad K' = (S', T', F')$$

are the occurrence net representations of R and R' (see Definition 2.6), respectively, then the following is true: Whenever there is $a[i] \in S$, $a[j] \in S'$, and there is no $k > i$ such that $a[k] \in S$, and there is no $m < j$, such that $a[m] \in S'$, we can always (but not necessarily) delete $a[j+n]$ from R'', and replace the set $\{(a[j+n], y)\}$ by the set $\{(a[i], y)\}$.

R'', as modified above, is also a NP/R net over Σ. It is called the *concatenation* of R and R'. We call R a *predecessor* of R', and R' a *successor* of R.

Definition 5.5. Let R be a NP/R net. Denote the set of all predecessors of R with PRED(R), and the set of all successors of R with SUC(R). Note that either of these may be empty, and SUC(R) is only defined when R is finite.

Let PRS_1 and PRS_2 be the sets of all NP/R nets over the two AC/EN systems Σ_1 and Σ_2, respectively. The relation

$$BS \subseteq 2^{PRS_1} \times 2^{PRS_2}$$

is called the *b-equivalent relation*, or *bisimulation*, if

(a) for $(PR_1, PR_2) \in BS$ the following holds
- $\forall R_1 \in PR_1$, R_1 has finitely many pages,
- $\exists R_2 \in PR_2$, R_2 has finitely many pages,
- R_1 is b-isomorphic to R_2, and
- $(\text{SUC}(R_1), \text{SUC}(R_2)) \in BS$; and

(b) $(PR_1, PR_2) \in BS$ implies $(PR_2, PR_1) \in BS$.

Definition 5.6. Let Σ be an AC/EN system with M as its initial marking. The proper set of NP/R nets of Σ consists of all those NP/R nets which are exact simulations of some runs starting from M.

Theorem 5.7. *Let e_1 and e_2 be two CCS agents in normal form. Let $\Sigma(e_1)$ and $\Sigma(e_2)$ be the corresponding AC/EN systems. Then $(e_1, e_2) \in SB$ if and only if $(PR_1, PR_2) \in BS$, where PR_1 and PR_2 are the proper sets of NP/R nets of $\Sigma(e_1)$ and $\Sigma(e_2)$.*

In order to prove this, we need the following lemma.

Lemma 5.8. *Let e be an agent in normal form, $\Sigma(e)$ the corresponding AC/EN system (S, T, F, C, M). Then there is a one-to-one correspondence between the derivation sequences of e:*

$$e_i - H_i \rightarrow e_{i+1}, \quad 1 \leqslant i \leqslant n, \; e = e_1$$

and the firing sequences of $\Sigma(e)$:

$$M_i[B_i\rangle M_{i+1}, \quad 1 \leqslant i \leqslant n, \; M = M_1$$

where B_i is the step transforming M_i into M_{i+1}.

Proof. First, we prove the one-to-one correspondence between the derivations of e (one step derivation) and the firings of $\Sigma(e)$ (one step run). Let

$$e = \sum_{i=1}^{n} \prod_{j=1}^{n_i} e_{ij}$$

where each e_{ij} takes one of the following two forms:

$$\sum_{k=1}^{p} \alpha_k . e'_k \backslash A_k \,\text{where}\, x_k = e''_k \quad \text{or} \quad \textit{nil}.$$

Case 1. $n = 1$, i.e.

$$e = \prod_{j=1}^{n_1} e_{ij}.$$

Let $n_1 = m$, we differentiate two subcases.

Case 1.1. $m = 1$, i.e.

$$e = \sum_{k=1}^{p} \alpha_k . e_k \backslash A_k \text{ where } x_k = e_k''.$$

There is no need to consider the case where $e = nil$, because in that case there will be no action at all.

Clearly, $dec(e) = e$. From (61a) and (61d) in Definition 4.6 we know that the event of $\Sigma(e)$, which takes part in its first firing, must be one of the following:

$$e - \alpha_k \rightarrow dec(e_k[e_k''/x_k]) \text{ where } x_k = e_k'' \backslash A_k \quad \text{and } \alpha_k \notin A_k$$

This shows its one-to-one correspondence to the derivations of e.

Case 1.2. $m > 1$. Let

$$e = \prod_{j=1}^{m} e_j - H \rightarrow e',$$

then each e_j can take part in this derivation in one of the following three ways:

- e_j produces no action at all,
- e_j produces an action $\alpha \in H$,
- e_j communicates with another e_k and produces an action $\tau \in H$.

Now consider $\Sigma(e)$. Let

$$dec(e) = \{g_k \mid k = 1, \ldots, m\}, \qquad g_k = \prod_{j=1}^{k-1} id | e_k | \prod_{j=k+1}^{m} id.$$

It follows from (62e) and (62f) in Definition 4.7, that each g_k can take part in the firing of $\Sigma(e)$ in one of the following three ways:

- g_k is not a condition of any firing event;
- for some α, $g_k - \alpha \rightarrow g_k'$ fires independently (we call it simply α^k);
- g_k and some g_h, $h \neq k$, are both condition elements of the following firing event:

$$\{g_k, g_h\} - \tau \rightarrow \{g_k', g_h'\}$$

(we call it simply τ^{kh}).

These g_k's are different from each other. That means that in case $j \neq k, j \neq l, h \neq k, h \neq l$, the events α^k and α^j of $\Sigma(e)$ have no common condition elements. The same conclusion is true for α^k and τ^{jh}, or τ^{jh} and τ^{kl}.

It follows from (62e) and (62f) that

$$g_k' = \prod_{j=1}^{k-1} id | e_k' | \prod_{j=k+1}^{m} id.$$

Therefore, we have

$$\forall h \neq k, \ \forall x \in g_k', \ y \in g_h', \ x \neq y$$

because the numbers of id on the left and right sides of e_k' and e_h' are different.

Note that both g_k and g'_k are combined appropriately into binary products using parentheses. This means that the events α^k, τ^{hl}, etc. can fire concurrently in the sense of Definition 2.2. Thus we have shown the one-to-one correspondence between the event firings of $\Sigma(e)$ and the derivation of e.

Case 2. $n>1$.

Case 2.1. $\forall i, n_i=1$. This case is equivalent to Case 1.1.

Case 2.2. $\exists i$, $n_i>1$. Let

$$e=\sum_{i=1}^{n} e_i, \qquad e_i=\prod_{j=1}^{n_i} e_{ij}.$$

Then any action of e must be produced by some e_i. W.l.o.g. assume $n_i>1$. Let us recall Case 1.2. Let

$$G_{ij}=\sum_{h=1}^{i-1} dec(e_h)+\{g_{ij}\}+\sum_{h=i+1}^{n} dec(e_h)$$

where $g_{ij} \in dec(e_i)$:

$$g_{ij}=\prod_{k=1}^{j-1} id|e_{ij}|\prod_{k=j+1}^{n_i} id.$$

Note that $dec(e_{ij})=e_{ij}$ because e is in normal form. G_{ij} is the pre-condition of the event $G_{ij}-\alpha_{ij}\rightarrow G'_{ij}(=g'_{ij})$ or part of the pre-condition of the event $\{G_{ij}, G_{ik}\}-\tau\rightarrow\{G'_{ij}, G'_{ik}\}$ in $\Sigma(e)$, where α_{ij} is the action produced by e_{ij}, and τ is the action produced by e_{ij} and e_{ik}.

(i) $\forall j \neq k$, $g_{ij} \neq g_{ik}$. That means

$$\forall i, \forall j \neq k, \quad G_{ij} \cap G_{ik}=\emptyset.$$

This shows that the events α^{ij} or τ^{ijk} in $\Sigma(e)$, which correspond to the actions α_{ij} or τ mentioned above, have no pre-condition elements in common.

(ii) We have

$$g'_{ij}=\prod_{k=1}^{j-1} id|dec(e'_{ij})|\prod_{k=j+1}^{n_i} id$$

where $e_{ij}-\alpha\rightarrow dec(e'_{ij})$. Therefore

$$\forall j \neq k, \forall x \in g'_{ij}, \quad y \in g'_{ik}, \quad x \neq y$$

and

$$G'_{ij} \cap G'_{ik}=\emptyset.$$

This shows that the events α^{ij} or r^{ijk} have no post-condition elements in common. It follows then that these events can fire concurrently in $\Sigma(e)$, by combining (i) and (ii).

(iii) $\forall l \neq j, \forall h, k$

$$G_{lh} \cap G_{jk} \neq \emptyset.$$

This shows that the events of $\Sigma(e)$, which correspond to actions of different e_i's cannot fire concurrently.

Now we can conclude by combining (i)–(iii) that in Case 2.2 the event firings within a step in $\Sigma(e)$ are in a one-to-one correspondence to the one step derivations of e.

It is now easy to prove the conclusion by induction for a derivation sequence consisting of any number of steps. □

Proof of Theorem 5.7. Let $(e_1, e_2) \in SB$, $R_1 \in PR_1$, with finitely many pages. Then R_1 must be an exact simulation of a run A_1 of $\Sigma(e_1)$, starting from its initial marking M_1. Assume that A_1 consists of the steps $B_{11}, B_{12}, \ldots, B_{1n}$. It follows from Lemma 5.8 that A_1 corresponds to a derivation sequence of e_1

$$(e_1 =)\ e_{11} \rightarrow e_{12} \rightarrow e_{13} \rightarrow \cdots \rightarrow e_{1,n+1}. \tag{67}$$

It corresponds by bisimulation to the following derivation sequence

$$(e_2 =)\ e_{21} \rightarrow e_{22} \rightarrow e_{23} \rightarrow \cdots \rightarrow e_{2,n+1}. \tag{68}$$

By Lemma 5.8, this corresponds again to a run A_2 of $\Sigma(e_2)$, which consists of steps $B_{21}, B_{22}, \ldots, B_{2n}$. Let R_2 be a NP/R net which is an exact simulation of A_2 and is constructed according to the way of Theorem 2.16. Clearly R_2 is finite.

At first, we will show that R_1 is b-isomorph to R_2. Let (S^1, T^1, F^1) and (S^2, T^2, F^2) be occurrence representations of R_1 and R_2 as it is prescribed in Definition 2.6(a). Then it follows from Definition 2.15 and Theorem 2.16, that there exist two bi-unique mappings

$$\beta_1 : ev(A_1) \rightarrow T^1, \qquad \beta_2 : ev(A_2) \rightarrow T^2.$$

From Lemma 5.8 and the bisimulation relation between e_1 and e_2 we know that there is also a one-to-one correspondence between event occurrences of $ev(A_1)$ and $ev(A_2)$. Denote it by ψ, then

$$\beta_2 \circ \psi \circ \beta_1^{-1} : T^1 \rightarrow T^2$$

is a one-to-one correspondence between events of R_1 and R_2.

The event occurrences x of $ev(A_1)$ take the form of $(G - \alpha \rightarrow G'\langle i \rangle)$. From Corollary 2.7 we know that the $\beta_1(x)$'s take the form $(G - \alpha \rightarrow G')[j]$. Similar conclusions are valid for $ev(A_2)$ and β_2. Then, again based on Lemma 5.8 and bisimulation of e_1, e_2, we conclude

$$\beta_2 \circ \psi \circ \beta_1^{-1}((G_1 - \alpha \rightarrow G'_1)[i]) = G_2 - \alpha \rightarrow G'_2[j]$$

where $G_u - \alpha \rightarrow G'_u$ is an event of $\Sigma(e_u)$ $(u = 1, 2)$. By taking 3) of Definition 2.15 into consideration, we know that for arbitrary $t_1, t'_1 \in ev(A_1)$ the following is true:

$$\beta_1(t_1) \prec \beta_1(t'_1) \Leftrightarrow \beta_2 \circ \psi(t_1) \prec \beta_2 \circ \psi(t'_1),$$

where $\Leftrightarrow$ means "if and only if".

This shows that R_1 is b-isomorph to R_2.

Now assume that R'_1 is a successor of R_1 and has finitely many pages. Assume further that R''_1 is the concatenation of R_1 and R'_1. From Definition 5.4 we know that R''_1 belongs to the proper set of NP/R nets over $\Sigma(e_1)$. It corresponds to a run A''_1 of $\Sigma(e_1)$, namely,

$$B_{11}, B_{12}, \ldots, B_{1n}, B_{1,n+1}, \ldots, B_{1,m}, \quad m>n$$

and thus to a derivation sequence of e_1:

$$(e_1 =)\ e_{11} \to e_{12} \to e_{13} \to \cdots \to e_{1,n+1} \to \cdots \to e_{1,m}$$

and thus to a derivation sequence of e_2:

$$(e_2 =)\ e_{21} \to e_{22} \to e_{23} \to \cdots \to e_{2,n+1} \to \cdots \to e_{2,m}$$

and thus to a run A''_2 of $\Sigma(e_2)$:

$$B_{21}, B_{22}, \ldots, B_{2n}, B_{2,n+1}, \ldots, B_{2,m}$$

Note that A_2 is a subrun of A''_2.

Construct a NP/R net R'_2 which is an exact simulation of the step sequence $B_{2,n+1}, \ldots, B_{2,m}$ by using the algorithm given in Theorem 2.16. R'_2 has also finitely many pages. Based on the same way of reasoning we can show that R'_2 is b-isomorph to R'_1. Let

$$\Sigma(e_1) = (S_1, T_1, F_1, C_1, M_1),$$

$$\Sigma(e_2) = (S_2, T_2, F_2, C_2, M_2);$$

where

$$(M_1 =)\ M_{11}[B_{11}\rangle M_{12} \cdots M_{1n}[B_{1n}\rangle M_{1,n+1},$$

$$(M_2 =)\ M_{21}[B_{21}\rangle M_{22} \cdots M_{2n}[B_{2n}\rangle M_{2,n+1},$$

$$\Sigma'(e_1) = (S_1, T_1, F_1, C_1, M_{1,n+1}),$$

$$\Sigma'(e_2) = (S_2, T_2, F_2, C_2, M_{2,n+1}).$$

Apply the same proof procedure which has been applied to $e_1, e_2, \Sigma(e_1), \Sigma(e_2)$, to $e_{1,n+1}, e_{2,n+1}, \Sigma'(e_1), \Sigma'(e_2)$, we can obtain similar conclusions. Continue this process for arbitrary n and we complete the proof of the theorem in one direction. The proof of the other direction is similar and is thus omitted here. □

Corollary 5.9. *Let e_1 and e_2 be two arbitrary CCS agents, $N(e_1)$ and $N(e_2)$ be their normal forms (not necessary unique), $\Sigma(N(e_1))$ and $\Sigma(N(e_2))$ be the corresponding AC/EN systems. Then $(e_1, e_2) \in SB$, if and only if $(PR_1, PR_2) \in BS$, where PR_1 and PR_2 are the proper sets of NP/R nets over $\Sigma(N(e_1))$ and $\Sigma(N(e_2))$, respectively.*

Acknowledgment

The work of using Petri nets to describe the true concurrency semantics of CCS was introduced to me by Prof. Montanari while I was visiting the University of Pisa. I owe many thanks to him and his research group for this inspiration.

References

[1] P. Degano, R. De Nicola and U. Montanari, CCS is an (augmented) contact free C/E system, in: *Proc. Advanced School on Mathematical Models for the Semantics of Parallelism*, Lecture Notes in Computer Science, Vol. 280 (Springer, Berlin, 1987) 144–165.

[2] Lu Ruqian, P/R nets and process concepts (I), (II), *Acta Scientica* **35**(1) (1992) 21–31; **35**(2) (1992) 148–157.

[3] R. Milner, *A Calculus of Communicating Systems*, Lecture Notes in Computer Science, Vol. 92 (Springer, Berlin, 1984).

[4] C.A. Petri, *Non-Sequential Processes*, GMD, ISF 77-05

[5] W. Reisig, *Petri Nets* (Springer, Berlin, 1985).