LFCS

Laboratory for Foundations of Computer Science
Department of Computer Science - University of Edinburgh

Self-independent Petri Nets (or a dead-lock-free paradigm)

by

SUN, Yong

LFCS Report Series

ECS-LFCS-89-98
(also published as CSR-316-89)

November 1989

LFCS
Department of Computer Science
University of Edinburgh
The King's Buildings
Edinburgh EH9 3JZ

Self-independent Petri Nets
(or a dead-lock-free paradigm)

SUN, Yong*

briefly revised November 1989†

Brief Overview

As we know, hand-shaking is an abstract and ideal synchronous mechanism for communication. However, it has implementation problems in distributed systems, so-called synchronization problem (see [Lamport 82, Sun 87] for references), i.e. asynchronous mechanisms for communication are inevitable in implementation. So, we are considering to bring such asynchrony into semantic models for concurrency directly, instead of the common practice of simulating the asynchrony through the hand-shaking mechanism

The second consideration of ours is dead-lock problem in concurrency, which is closedly associated to synchrony (or hand-shaking). For instance, $\alpha?x.\beta?y.B(x,y)||\beta!e_1.\alpha!e_2.C$ *in CSP and* $(\alpha x.\beta y.B(x,y)|\bar{\beta}e_1.\bar{\alpha}e_2.C)\backslash\alpha\beta$ *in CCS are examples of dead-locked processes and of dead-locked agents respectively. Such instances should be avoided, if possible, in implementation of distributed (or communicating) systems. Obviously, if there exists a (complete) inference system which can be used to decide whether there is a dead-lock in a communicating system, then we can use such an inference system to decide and to exclude the dead-lock instances. But, this kind of decision problem for dead-locks is generally undecidable. Therefore, we have to try another way around, i.e. a dead-lock-free paradigm. This paper pursuits such a dead-lock-free paradigm to certain extent.*

The third consideration in our mind is to present a denotational model for CCS in a non-interleaving fashion (or so-called true concurrency), although the actual interpreting CCS terms into the model is omitted. It is just a matter of fact that the interested readers can follow a quite standard procedure and provide the corresponding interpretations. In case of any doubt, you can consult Goltz and Mycroft's paper [Goltz 84] or Winskel's [Winskel 83].

The model to be presented can be viewed essentially as is constructed from labelled Petri-nets. All constructible nets are self-independent (or dead-lock-free). Technically, the result can be regarded as an improvement of (a) Winskel's event structures [Winskel 83] which semantics is non-symmetricity to non-determinism (i.e. the semantics of $B_1 + B_2$ *is not equal to semantics of* $B_2 + B_1$*); of (b) Goltz and Mycroft's labelled Petri-nets [Goltz 84] which can not systematically deal with recursive processes (i.e. they have a trouble in giving a semantics to* B_2 *in* $\{B_1 \Leftarrow$

*LFCS, Department of Computer Science, University of Edinburgh, King's Buildings, Mayfield Road, Edinburgh EH9 3JZ (U.K.).

†This paper is a briefly revised working note on congruences of labelled Petri Nets in May 1987; which was once presented in Edinburgh Concurrency Club's seminars in 1987.

$\alpha.Nil|\gamma.B_1, B_2 \Leftarrow B_1|\beta.B_2\}$), although isomorphic nets are treated as identical; and of (c) the work of Boudol and Castellani in [Boudol 87] which follows Winskel's approach and solve the problem of non-symmetricity of non-determinism, but can only discuss finite structures. Therefore, this paper provides an appropriate and sufficiently abstract domain of Petri Nets for solving fixpoints of semantic equations when isomorphic nets are regarded as identical.

Contents

1 Introduction to Self-independent Petri Nets (SPNs)

A labelled Petri Net (LPN or simply PN) N can be denoted as $< S, E, T, L >$, where S is a set of states (or places), E is a set of events, T is a set of transitions which is a subset of $(S \times E) \cup (E \times S)$, L is a labelling from events E to labels Lab. Later, if no confusion, we let N be $S \cup E$.

For $x \in N$, the pre-set of x and the post-set of x are defined respectively as (i) ${}^\bullet x =_{df} \{y \in N | < y, x > \in T\}$ and as (ii) $x^\bullet =_{df} \{y \in N | < x, y > \in T\}$. A root-set of net N is ${}^\circ N =_{df} \{x \in N | {}^\bullet x = \emptyset\}$. $|X|$ stands for the cardinal number of a set X. Roots of N (i.e. elements in ${}^\circ N$) can be considered of always initially marked, although we do not mention markings at all in this paper. This treatment can be viewed as presupposing both that all roots (states) are always marked initially and that multi-tokens are linearized into multi-nets, i.e. every initial state represents one token (see an example in figure 1.1).

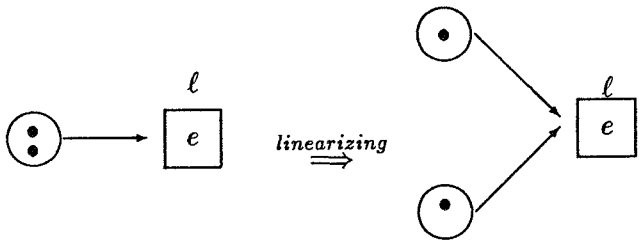

figure 1.1

For Petri Nets, there are three phenomena need to be classified. (a) events e_1 and e_2 in figure 1.2 are to be exclusively triggered, i.e. either can be triggered but not both. In other words, these two events are not independent of each other. This phenomenon is commonly referred to as non-determinism.

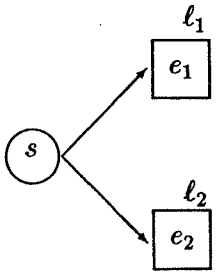

figure 1.2

(b) states s_1 and s_2 in figure 1.3 are independent of each other, i.e. both states s_1 and s_2 are triggered by event e at a same time. This phenomenon is commonly referred to as concurrency or parallelism.

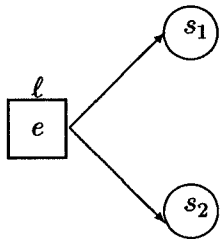

figure 1.3

(c) event e in figure 1.4 can only be triggered by both states s_1 and s_2 not only one of them. This phenomenon is commonly referred to as synchronization.

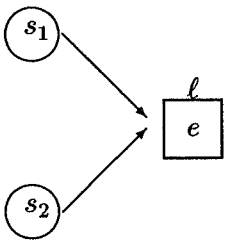

figure 1.4

To exploit these phenomena further, we introduce an parallel operator | over nets. Informally, let N_1 be the net (see figure 1.5) :

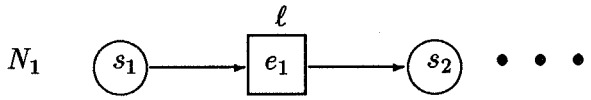

figure 1.5

and N_2 be (see figure 1.6) :

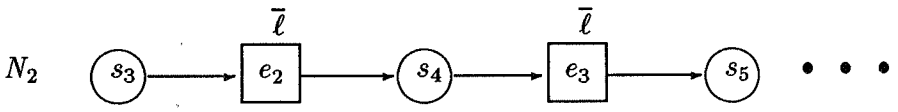

figure 1.6

Their composition $N_1|N_2$ is (see figure 1.7) :

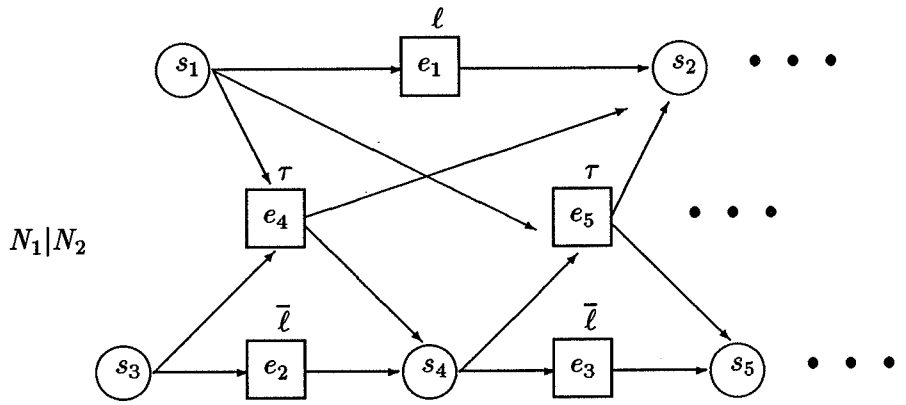

figure 1.7

In these pictures (see figure 1.5, figure 1.6 and figure 1.7), each horizontal arrow can roughly be viewed as an execution (or evolution) of a process, and the arrows between horizontal arrows are interactions between processes. The events labelled by τ's (say e_4 and e_5) are successful communicating actions, and events labelled by non-τ's (say e_1 and e_2) means unsuccessful communicating actions. More specifically, N_1 of $N_1|N_2$ does not know whether the action performed on port ℓ is a result of event e_1 or a result of event e_4 (or e_5), if it does not check the receiving message. That is, the result of an action has to be examined before N_1 understand whether it is a successful action. In other words, an empty message implies that the action was unsuccessful. Among successful actions, N_1 need to be informed from N_2 to know which event resulted in the received message, event e_4 or event e_5. Therefore, every communicating action is only subject to the performer's wish (or will) regardless (i) who he is (a message sender or a message receiver) and (ii) his communicating partner. Synchronization is a result of coincidence or was established by protocols, therefore, synchrony is a special case of asynchrony. Nevertheless, we obtain some remarkable power from such asynchrony. We name two of them : (a) every process is *self-independent*; and (b) message loss in communication is a natural consequence of the asynchronous communication, therefore tolerance of message loss is naturally inherited of the model, which is an important issue (fault-tolerance) in distributed computing systems.

Actually, we say that a relation $||$[1] on nets is the *independent* relation on nets[2], i.e. $|| : (S \times S) \cup (E \times E) \cup (S \times E) \cup (E \times S)$ is the least relation on net N which satisfies the following six conditions.

(1) $\forall s_1, s_2 \in {}^\circ N.\ s_1 \neq s_2 \Rightarrow s_1 || s_2$
(2) $\forall e \in E, \forall s_1, s_2 \in e^\bullet.\ s_1 \neq s_2 \Rightarrow s_1 || s_2$
(3) $\forall e_1, e_2 \in E.\ ({}^\bullet e_1 \cap {}^\bullet e_2 = \emptyset)\ \&\ ((\forall s_1 \in {}^\bullet e_1, \forall s_2 \in {}^\bullet e_2.\ s_1 || s_2) \Rightarrow e_1 || e_2)$
(4) $\forall e_1, e_2 \in E.\ e_1 || e_2 \Rightarrow (\forall s_1 \in {}^\bullet e_1.\ s_1 || e_2)\ \&\ (\forall s_2 \in {}^\bullet e_2.\ e_1 || s_2)$
(5) $\forall s \in S, \forall e \in E.\ (\forall s' \in {}^\bullet e.\ s || s') \Rightarrow s || e\ \&\ e || s$
(6) $\forall s \in S, \forall e \in E.\ (s || e \Rightarrow \forall s' \in e^\bullet.\ s || s')\ \&\ (e || s \Rightarrow \forall s' \in e^\bullet.\ s' || s)$.

Intuitively, the first condition (1) says that distinct initial states are independent of each other; that the distinct states triggered by a same event are independent of each other is said by the second (2); the third (3) expresses that distinct events triggered by totally independent states are independent of each other; the fourth (4) indicates that an event and a state are independent

[1] Do not confuse this symbol with parallel operator || in CSP, although they are closely related as well.
[2] Or more precisely, on *well-rooted* Nets. The concept of well-rooted will be defined later.

of each other if the state is triggered by another event independent of the event; The fifth (5) suggests that an event and a state are independent of each other if all states triggering the event are independent of the state; and the last one (6) shows that a state is independent of another state if the other state is triggered by an event which is independent of the state.

With the understanding of independence relations on nets, we are interested in the nets which have the following six properties

(i) $\forall x, y \in E \cup S.\ {}^\bullet x = {}^\bullet y\ \&\ x^\bullet = y^\bullet \Rightarrow x = y$

(ii) $\forall e \in E, \exists s_1, s_2 \in S.\ s_1 T e\ \&\ e T s_2$

(iii) $||$ is irreflexive.

(iv) $\forall e \in E, \forall s_1, s_2 \in {}^\bullet e.\ s_1 \neq s_2 \Rightarrow s_1 || s_2$,

(v) $\forall s \in S, \forall e_1, e_2 \in {}^\bullet s.\ \neg(e_1 || e_2)$.

(vi) $\forall s \in S - {}^\circ N.\ \bigcap\{{}^\bullet e | e \in {}^\bullet s\} \neq \emptyset$

Intuitively, the first property (i) says that the nets we are interested in have no redundancy; the second (ii) expresses that every event has a cause and a result in the nets we are inetersted; the third (iii) assures of that one is impossible to be independent of itself; The fourth (iv) suggests that distinct states triggering a same event must be independent of each other; The distinct events triggering a same state can not be independent of each other is expressed by the fifth (v); and the last (vi) asserts that if a non-initial state triggered by a collection of events then the events must share a common triggering state.

For any net N, if the independence relation $||$ on the net have the above six properties, then we say that N is a *Self-independent* (Labelled) Petri Net, a SPN for short. Actually, property (iv) of SPNs excludes the instance of nets in figure 1.8, and property (v) excludes the instance in figure 1.9[3].

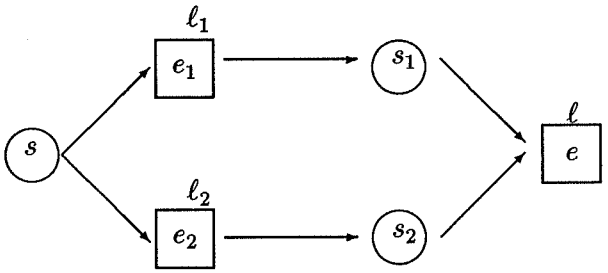

figure 1.8

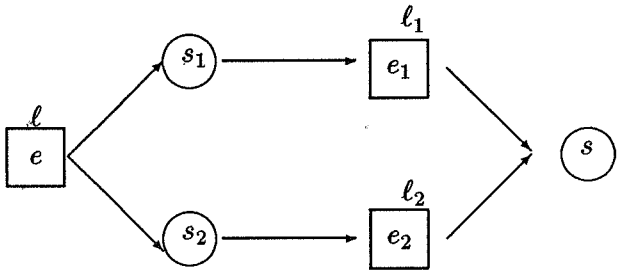

figure 1.9

[3]The nets in figure 1.8 and figure 1.9 are co-nets of each other, i.e. one is obtained from the other by reversing all transitions.

Intuitively, these exclusions are very reasonable, i.e. (a) *non-determinism* should not lead *synchronization* and (b) *concurrency* (or parallelism) should not reduce to *non-determinism*.

From now on, we concentrate on SPNs. Also, it is worth to mention that $||$ is a kind of reversed version of $\#$ (conflict relation) in [Neilsen 79], but the nets involved in this paper are richer than theirs, since our nets are not necessarily cycle-free. However, the existence of a circle in a net is closely related to dead-lock in the net. An example with a circle is shown in figure 1.10. Fortunately, this does not happen in the constructible nets of this paper[4]. Thus, it explains the reason for the alternative title of this paper.

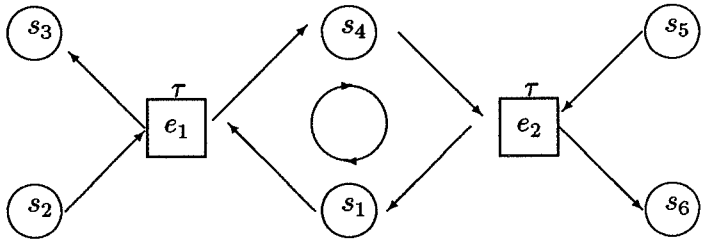

figure 1.10

Apparently, we are not interested in a trivial SPN, i.e. $< \emptyset,\emptyset,\emptyset,\emptyset >$ is not inside the SPNs under our consideration. Also, since we intend to regard isomorphic nets as identical, we identify all singleton nets as one and write it as $< \{\bullet\},\emptyset,\emptyset,\emptyset >$. For simplicity, we further assume that roots of each net are states, i.e. ${}^\circ N \subseteq S$. Actually, this assumption is implicitly implied by the property (ii) of self-independent nets. Then, we introduce a partial order $\sqsubseteq$ on SPNs as that $N_1 \sqsubseteq N_2$ iff the following three conditions hold :

(a) $N_1 \subseteq N_2$ (or $S_1 \subseteq S_2$, $E_1 \subseteq E_2$, $T_1 \subseteq T_2$, and $L_1 \subseteq L_2$),

(b) ${}^\circ N_1 \subseteq {}^\circ N_2$, and

(c) $\forall x, y \in N_1 . x\, T_{N_2} y \supset x\, T_{N_1} y$.

For the third condition (c), it implies $N_2 \lceil_{N_1} = N_1$. Intuitively it means that the prefix closure implies the partial order. Hence, the least upper bound $\sqcup N_k$ of an ω-chain $\{N_k\}$ is $\sqcup N_k = \bigcup N_k$, and the least element $\perp$ is $< \{\bullet\},\emptyset,\emptyset,\emptyset >$. Therefore, SPNs form a *cpo*.

2 Constructible SPNs

Let ℓ range over Lab, ℓ and $\overline{\ell}$ represent input and output respectively, $^-$ be the *complementary* function on Lab. We are giving constructions for SPNs below.

(i) (bottom) $\perp =< \{\bullet\},\emptyset,\emptyset,\emptyset >$;

(ii) (sequence) $\ell.N =_{df} < \{s\} \cup S, \{e\} \cup E, \{< s,e >\} \cup \{< e,s' > | s' \in {}^\circ N\} \cup T, L \cup \{< e,\ell >\} >$ where $N \neq \emptyset$, and s, e are not in N;

(iii) (non-deterministic composition) $N_1 + N_2 =_{df} < S, E_1 \cup E_2, T, L_1 \cup L_2 >$,

where $N_i \neq \emptyset$ $(i \neq 1,2)$ and disjoint with each other, $S =_{df} (S_1 \backslash {}^\circ N_1) \cup (S_2 \backslash {}^\circ N_2) \cup ({}^\circ N_1 \times {}^\circ N_2)$ and $T =_{df} \{< s,e > | s =< s_1,s_2 > \wedge (< s_1,e > \in T_1 \vee < s_2,e > \in T_2) \wedge s_i \in {}^\circ N_i (i = 1,2)\} \cup T_1 \lceil_{(S_1 \backslash {}^\circ N_1) \times E_1 \cup E_1 \times (S_1 \backslash {}^\circ N_1)} \cup T_2 \lceil_{(S_2 \backslash {}^\circ N_2) \times E_2 \cup E_2 \times (S_2 \backslash {}^\circ N_2)}$;

(iv) (parallel composition) $N_1 | N_2 =_{df} < S_1 \cup S_2, E_1 \cup E_2 \cup prod, T_1 \cup T_2 \cup conc, L_1 \cup L_2 \cup comm >$,

[4]There are some subtle points about synchronizers, see next section for the definition of synchronizers.

where N_1 and N_2 disjoint with each other, and $prod =_{df} \{e \in E_1 \times E_2 | L(e\lceil_{E_1}) = \overline{L(e\lceil_{E_2})} \neq \tau\}$, $comm =_{df} \{< e, \tau > | e \in prod\}$, $conc =_{df} \{< s_1, e >, < s_2, e >, < e, s^1 >, < e, s^2 > | e \in prod \wedge s_1 \in {}^\bullet(e\lceil_{E_1}) \wedge s_2 \in {}^\bullet(e\lceil_{E_2}) \wedge s^1 \in (e\lceil_{E_1})^\bullet \wedge s^2 \in (e\lceil_{E_2})^\bullet\}$;

(v) (synchronizer) $N\backslash\ell$ will be defined later;

(vi) (recursive operator) μ or fix (fixpoint operator).

Now, to motivate the definition of synchronizer(say $N\backslash\ell$), we provide an example for it. Let us look at the example of $N_1|N_2$ of figure 1.7 in previous section. The intuitive meaning of $(N_1|N_2)\backslash\ell$ is to force that all communications on port ℓ must be synchronized. The effect of such enforcement is demonstrated in figure 2.1.

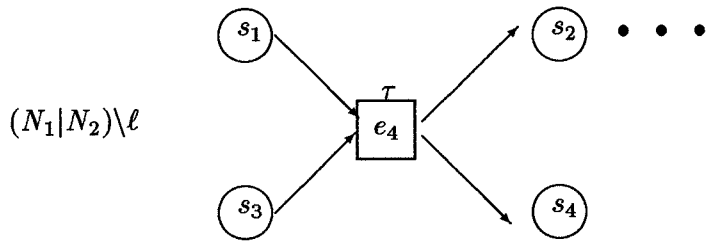

figure 2.1

Apparently, by referring to figure 1.7, we know that events e_1 and e_2 are cut off after synchronization in figure 2.1. Also, there is no event which follows state s_4, i.e. events e_3 and e_5, and state s_5 are cut off as well. In a contrast, there may be events which follows state s_2. In order to formally capture such synchronization, we have to introduce another concept, so-called *well-rooted* SPNs.

Let $N^{[0]} =_{df} {}^\circ N (\subseteq S)$, $N^{[2i+1]} =_{df} \{e \in E | {}^\bullet e \subseteq \bigcup_{k=0}^{i} N^{[2k]}\}$, $N^{[2i+2]} =_{df} \{s \in S | {}^\bullet s \cap N^{[2i+1]} \neq \emptyset\}$, and $N^* =_{df} < \bigcup N^{[2i]}, \bigcup N^{[2i+1]}, T\lceil_{(\bigcup N^{[2i]}) \times (\bigcup N^{[2i+1]}) \cup (\bigcup N^{[2i+1]}) \times (\bigcup N^{[2i]})}, L\lceil_{\bigcup N^{[2i+1]}} >$. We say that a SPN N is *well-rooted* iff $N = N^*$.

Informally, the idea of well-rooted nets is similar to the idea of well-known property of set theory, i.e. well-foundness. The *well-rooted* N means that if $x \in N$, then x can be reached by an (arbitrary) finite number of steps (or executions, evolutions) from the roots ${}^\circ N$. In other word, a well-rooted N says that every state in N is reachable. Nevertheless, we are only interested in the well-rooted SPNs in this paper.

Lemma 2.1 : The property of well-rooted is an invariant of the constructions of prefixs, non-deterministic composition, and parallel compositions.

For a well-rooted N, we define $N\backslash\ell =_{df} < S', E', T\lceil_{S' \times E' \cup E' \times S'}, L\lceil_{E'} >$, where $N^{(0)} =_{df} {}^\circ N$, $N^{(2i+1)} =_{df} \{e \in E | {}^\bullet e \subseteq \bigcup_0^i N^{(2k)} \wedge L(e) \neq \ell, \bar{\ell}\}$, $N^{(2i+2)} =_{df} \{s \in S | {}^\bullet s \cap N^{(2i+1)} \neq \emptyset\}$, $S' =_{df} \bigcup N^{(2i)}$, and $E' =_{df} \bigcup N^{(2i+1)}$. Thus, the result of Lemma 2.1 can be extended to include synchronizers.

Lemma 2.2 : The well-rooted property is an invariant of the constructions mentioned in Lemma 2.1 with extra constructions of synchronizers.

Also, we have that the well-rooted property is closed under the least upper bounds. Formally

Lemma 2.3 : Let $\{N_i\}$ be an ω-chain of SPN, if N_i is well-rooted for all i, then $\sqcup N_i$ is well-rooted, too.

Therefore, the result of Lemma 2.2 can be extended to include all constructions of nets.

Theorem 2.4 : The well-rooted property is an invariant of all constructions of nets.

The key point of the proof for Theorem 2.4 is that all constructions are monotonic and continuous. So, fixpoints always exist when we apply μ (or fix) to all possible combinations of constructions, and others would naturally follows from previous lemmas.

It is not hard to verify that all constructible nets are Self-independent. Further, if we regard every root of a net as a process and if we syntactically exclude the instance of constructions like $N_1 + (N_2|N_3)$, then every process is self-independent in constructible nets. That is, every token can move along constructible nets without getting stucked if we view that each root is always initially marked. This is where the name of *Self-independent Petri Nets* comes from.

However, there are some subtle points about synchronizers with regard to self-independency. Nevertheless, we deliberately omit them in the present paper.

3 Isomorphic Classes of SPNs ([SPN])

To regard non-deterministic compositions of $N_1 + N_2$ and of $N_2 + N_1$ to be identical, we have to consider isomorphic classes of SPNs. First of all, let us give the definition for net isomorphism. N_1 is *isomorphic to* N_2 ($N_1 \cong N_2$) iff there is a bijection $\phi : N_2 \to N_1$ such that $s \in S_2 \Leftrightarrow \phi(s) \in S_1$, $e \in E_2 \Leftrightarrow \phi(e) \in E_1$, $< x, y > \in T_2 \Leftrightarrow < \phi(x), \phi(y) > \in T_1$ and $L_2 = L_1 \circ \phi$. It can easily be checked that the isomorphic relation among SPNs is an equivalence relation among them. Then, we define the corresponding equivalent class as : $[N] =_{df} \{N' \in SPN | N' \cong N\}$.

Later we refer to the set of all equivalent classes as [SPN].

Lemma 3.1 : The following three definitions are equivalent : for all N_i ($i = 1, 2$),

1. $[N_1] \trianglelefteq_1 [N_2]$ iff $\exists N_2' \in [N_2].N_1 \sqsubseteq N_2'$,
2. $[N_1] \trianglelefteq_2 [N_2]$ iff $\exists N_1' \in [N_].N_1' \sqsubseteq N_2$,
3. $[N_1] \trianglelefteq_3 [N_2]$ iff $\exists N_1' \in [N_1], \exists N_2' \in [N_2].N_1' \sqsubseteq N_2'$.

Next, we naturally want to extend the partial order $\sqsubseteq$ from SPN to [SPN], and denote it as $\trianglelefteq$. Because of Lemma 3.1, we can refer the order $\trianglelefteq$ to any one of $\trianglelefteq_i$ ($i = 1, 2, 3$) as we wish. However, $\trianglelefteq$ is not necessarily a partial order in general but a pre-order. The problem not to be a partial order comes from symmetricity, i.e. $[N_1] \trianglelefteq [N_2] \trianglelefteq [N_1]$ does not imply $[N_1] = [N_2]$.

In order to present the non-partial-order problem more clearly, let $\preceq$ be a partial order in an arbitrary X and $\sim$ be an isomorphism on X. Then we say that a partial order $\preceq$ in X has a *well-extended* property (on X over $\sim$) if

$$\forall x_1, x_2 \in X. x_1 \preceq x_2 \wedge x_1 \cong x_2 \supset x_1 = x_2.$$

It is obvious that $\trianglelefteq$ is a partial order if $\sqsubseteq$ has the well-extended property. Unfortunately, $\sqsubseteq$ does not have the well-extended property in general. Therefore, from this it is not hard to conclude that $\trianglelefteq$ can not be a partial order in SNPs in general. An example to show that $\sqsubseteq$ does not have the well-extended property is given as follows, see figure 3.1[5].

Let $N_i =< S_i, E_i, T_i, L_i >$ ($i = 1, 2$) and

(a) $S_i = \{s^i\} \cup \{s^i_{n,k} | ((i = 1) \wedge (n \geq 1) \wedge (1 \leq k \leq 2n)) \vee ((i = 2) \wedge (n \geq 1) \wedge (1 \leq k \leq 2n+1))\}$;

(b) $E_i = \{e^i\} \cup \{e^i_{n,k} | ((i = 1) \wedge (n \geq 1) \wedge (1 \leq k \leq 2n-1)) \vee ((i = 2) \wedge (n \geq 1) \wedge (1 \leq k \leq 2n))\}$;

[5] An example of such was pointed out to me by G. Plotkin during one of my presentations to Concurrency Club's seminar in 1987, which is obtained by taking the prefix away from figure 3.1.

(c) $T_i = \{< s^i, e^i >\} \cup \{< e^i, s^i_{n,1} > | n \geq 1\} \cup \{< s^i_{n,k}, e^i_{n,k} >, < e^i_{n,k}, s^i_{n,k+1} > | ((i = 1) \wedge (n \geq 1) \wedge (1 \leq k \leq 2n - 1)) \vee ((i = 2) \wedge (n \geq 1) \wedge (1 \leq k \leq 2n))\}$;

(d) $L_i = \{< e^i, \alpha >\} \cup \{< e^i_{n,k}, \beta > | ((i = 1) \wedge (n \geq 1) \wedge (1 \leq k \leq 2n - 1)) \vee ((i = 2) \wedge (n \geq 1) \wedge (1 \leq k \leq 2n))\}$.

It is obvious that N_1 is not isomorphic to N_2. But we still have to check whether there exist N_1' and N_2' as required in $\trianglelefteq_3$ of Lemma 3.1. This is done by considering an embedding N_1 into N_2, and conversely N_2 into N_1.

Therefore, in order to have $\trianglelefteq$ be a partial order, we need to restrict the SPNs to certain sub-collection of them. This sub-collection must be reasonably rich enough to accommodate CCS, say. The restriction we derive is the *finitely-branched* condition for SPNs. The following is to introduce the finitely-branched condition on SPNs.

Let N be a SPN. N is *finitely-branched* (or to be a fSPN) if N has the following two properties:

(i) for roots $|^\circ N| < \infty$;

(ii) $\forall e \in E.|e^\bullet| < \infty$; and

(ii) for events $\forall i.|N^{[2i+1]} - N^{[2i-1]}| < \infty$, where $N^{[-1]} = \emptyset$.

These three conditions of the finitely-branched conditions restrict SPNs to fSPNs such that the increase of the numbers of branches for each execution (or evolution) is limited to arbitrary finite numbers. Since a finite set is identical to another finite one if it is a subset of the latter and if there exists a 1-1 mapping between the elements of the two sets, fSPNs naturally have the well-extended property. We state the result as a lemma. Formally,

Lemma 3.2 : $\sqsubseteq$ **is well-extended in fSPNs.**

Therefore,

Theorem 3.3 : $\trianglelefteq$ **is a partial order in [fSPN].**

Now, we know that the finitely-branched condition is sufficient to guarantee that $\trianglelefteq$ being a partial order. But whether this condition is too restrictive or not (i.e. whether it is necessary or not) is to be demonstrated. However, we can show that it is not too restrictive in the sense that there are examples of which the symmetricity does not hold. Such an example has been given before in demonstrating whether the well-extended property holds in general. Nevertheless, we will give another one in the proof of Lemma 3.4. In this sense, we claim that the finitely-branched condition is a necessary condition for $\trianglelefteq$ being a partial order.

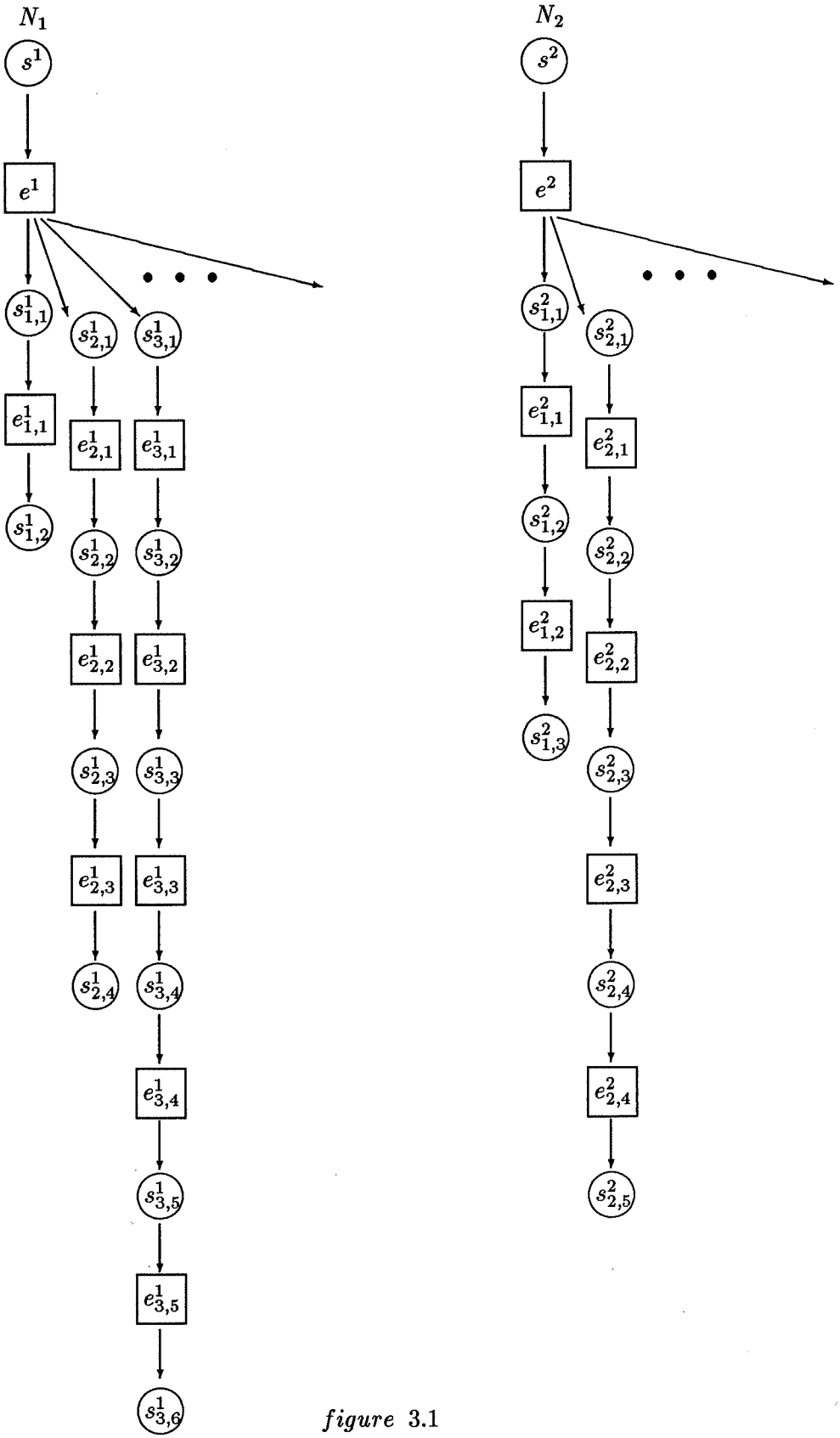
N_1
s^1
e^1
$s^1_{1,1}$
$s^1_{2,1}$
$s^1_{3,1}$
$e^1_{1,1}$
$e^1_{2,1}$
$e^1_{3,1}$
$s^1_{1,2}$
$s^1_{2,2}$
$s^1_{3,2}$
$e^1_{2,2}$
$e^1_{3,2}$
$s^1_{2,3}$
$s^1_{3,3}$
$e^1_{2,3}$
$e^1_{3,3}$
$s^1_{2,4}$
$s^1_{3,4}$
$e^1_{3,4}$
$s^1_{3,5}$
$e^1_{3,5}$
$s^1_{3,6}$
N_2
s^2
e^2
$s^2_{1,1}$
$s^2_{2,1}$
$e^2_{1,1}$
$e^2_{2,1}$
$s^2_{1,2}$
$s^2_{2,2}$
$e^2_{1,2}$
$e^2_{2,2}$
$s^2_{1,3}$
$s^2_{2,3}$
$e^2_{2,3}$
$s^2_{2,4}$
$e^2_{2,4}$
$s^2_{2,5}$

figure 3.1

Lemma 3.4 : There are $[N_1]$ and $[N_2]$ such that $\exists N_1' \in [N_1]$ and $\exists N_2' \in [N_2]$, they satisfy

1. $N_1 \sqsubseteq N_2'$, i.e. $[N_1] \trianglelefteq_1 [N_2]$,
2. $N_2 \sqsubseteq N_1'$, i.e. $[N_2] \trianglelefteq_1 [N_1]$, and
3. N_1 and N_2 are not isomorphic to each other, i.e. $[N_1] \neq [N_2]$.

Proof

As we point out previously, the example about the well-extended property can be used here. However, this only shows that $\forall e \in E.|e^\bullet| < \infty$ is necessary. Taken the prefix away from the example, it would turn out to be that $|^\circ N| < \infty$ is necessary. For the other case, we provide another example below, e.g. assume that $\exists j.|N^{[2j+1]} - N^{[2j-1]}| \not< \infty$ and $|N^{[2j+2]} - N^{[2j]}| \not< \infty$ (the conjunction is because of the nets' characteristics of no redundancy), say $j = 0$. The readers are recommended to draw a graphical representation of the following example, since a graph can help us to comprehend it at an intuitive level.

Let $N_i =< S_i, E_i, T_i, L_i >$ $(i = 1, 2)$, where

(a) $S_i = \{s^i\} \cup \{s_k^i | k \in Nat - \{0\}\}$;

(b) $E_i = \{e_{k,n}^i | ((i = 1) \wedge (1 \leq n \leq 2k - 1)) \vee ((i = 2) \wedge (1 \leq n \leq 2k))\}$;

(c) $T_i = \{< s^i, e_{k,n}^i >, < e_{k,n}^i, s_k^i > | ((i = 1) \wedge (1 \leq n \leq 2k - 1)) \vee ((i = 2) \wedge (1 \leq n \leq 2k))\}$; and

(d) $L_i = \{< e_{k,n}^i, \alpha^i > | ((i = 1) \wedge (1 \leq n \leq 2k - 1)) \vee ((i = 2) \wedge (1 \leq n \leq 2k))\}$.

It is obvious that N_1 is not isomorphic to N_2 (see figure 3.2). But we still need to check whether there exist N_1' and N_2' as required. This is done by considering an embedding N_1 into N_2, and conversely N_2 into N_1.

□

For an ω-chain $\{[N_k]\}$ in [fSPN], the least upper bound of it is : $\sqcup[N_k] = [\bigcup N_k']$ where $\forall k$. $N_k' \in [N_k] \wedge N_k' \sqsubseteq N_{k+1}'$. So, [fSPN] forms a *cpo*.

Naturally, We can extend the constructions in last section (section 2) to [fSPN], say $\alpha.[N] =_{df} [\alpha.N]$, $[N_1] + [N_2] =_{df} [N_1 + N_2]$, $[N_1]|[N_2] =_{df} [N_1|N_2]$ and $[N]\backslash\alpha =_{df} [N\backslash\alpha]$. Since the above constructions on [fSPN] are defined by their counterparts in fSPNs, their monotonicities and continuities follow easily. However, there is no trivial way to extend recursive definitions to isomorphic classes. For example, $x := x|\alpha.\perp$ where x is an identifier for nets; i.e. μ is not closed in fSPNs. So, we have to further restrict constructions to certain constructions. The answer we come up is the *well-constructed* restriction on net constructions.

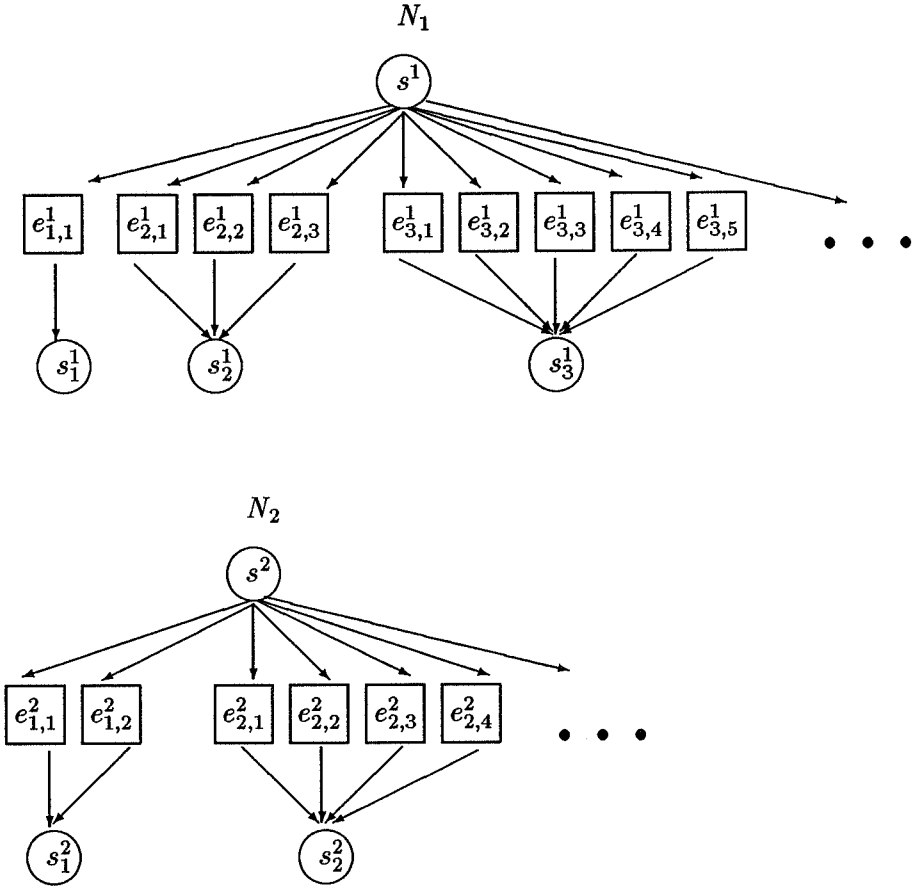

figure 3.2

An unary construction F (or functor) of fSPNs is said to be *well-constructed* if

$$\exists K.\forall k \geq K.|^\circ F^k([< \{\bullet\}, \emptyset, \emptyset, \emptyset >])| = |^\circ F^{k+1}([< \{\bullet\}, \emptyset, \emptyset, \emptyset >])| < \infty.$$

Intuitively the well-constructed condition for nets is to exclude the possibility of increasing the number of roots of a constructible net to an infinite number, i.e. a well-constructed F will not be able to increase the number of roots of a net to an arbitrary large number by recursively applying F. So, we have that

Theorem 3.5 : For any well-constructed F, $\mu x.(\lambda x.F)$ is closed in $[fSPN]$.

The proof is omitted. But, the key point of the proof of the above theorem is to show that a series of numbers of root states by recursively applying F (to the least element) is converged, and so, the fixed point of F belongs to fSPN.

A similar theorem of Theorem 3.5 for mutual recursive functors $\overline{F}$ is left out.

Since other constructions can easily be checked that they are closed in [fSPN], the net constructions in Section 2 under the well-constructed restriction are closed in [fSPN]. Therefore, it is safe to interpreting CCS terms into well-constructed [fSPN], i.e. the well-constructed [fSPN] is an adequet denotational semantics model for CCS provided the CCS terms are well-guarded as well.

4 Discussions

Looking from the categorical point, the constructions + and | agree with their sum and their product in the Petri nets category [Winskel 87] except that | is a restricted product which only affects the events with complementary labels. This shows that the intuition of the asynchrony from implementation has a categorical background. Also, it is worth mentioning that the asynchrony is dead-lock-free by nature. We should point out that the synchrony and the asynchrony in this paper are different from Milner's ones [Milner 84], where his synchrony means the existence of an universal clock in his framework and his asynchrony means the non-existence of such clock.

However, there is one construction does not fit our intuition of that every root of nets represents a process. This happens in the definition of +, i.e. it can produce more roots than the component's ones, even the sum of them. On the other hand, the idea behind + is non-determinism. So, it should not be able to produce more processes. Nevertheless, there is one possible way to overcome this by guarding | when + involves |. Intuitively, this solution is very reasonable. It means that *"non-determinism should not happen between a sequential process and concurrent processes"*, i.e. $N_1 + (N_2|N_3)$ should not be allowed in net constructions. If we accept this solution, then the constructible Petri Nets (or SPNs) in this paper would have a very good character, and they should be able to be axiomatized.

Another profit from the asynchrony not having been exploited in this paper is to incorporate broadcasting communication into our model. This is a subject of future investigation.

Comparing constructible nets (or cSPNs for short) with fSPNs, we understand that the collection of cSPNs is more restrictive than the one of fSPNs. In light of this observation, the condition for net constructions to be both of *finitely-branched* and of *well-constructed* may be over-restricted. This issue is certainly deserved our future attention. Nevertheless, it is always a good idea to bring semantical properties up to syntactical level. Recent Brown's work [Brown 89] has demonstrated that this is possible for Petri Nets through Linear Logic [Girard 86]. It also seems that a similar result can be achieved through operational semantics of nets (or transitional systems) and modal logic with certain extension of multiplicity.

5 Acknowledgement

Firstly, the author would like to thank the people who participated in Edinburgh Concurrency Club's seminar, and who provided many stimulations. It is hard to single out every benefit obtained. However, he sincerely thank G. Plotkin, who pointed out an example (see a footnote in section 3) to me during one of my presentations in the Club's seminars of 1987, which shows that $\sqsubseteq$ does not have the well-extended property in general. Also, he would like to thank W. Brauer for his detailed comment on an earlier version and his encouragement, to thank G. Winskel and A. Mycroft for their comment on an early draft of this paper. Thanks go to A. Knobel and P. Paczkowski for their proof-reading and comment on an earlier draft, which provide many stimulations on refining presentation of this paper.

6 References

Boudol 87 : G. Boudol and I. Castellani, "On the Semantics of Concurrency : Partial Order and Transition Systems", TAPSOFT '87, Lecture notes in computer science vol. 249, 1987.

Brown 89 : C. Brown, "Relating Petri Nets to Formulae of Linear Logic", LFCS report series No.87, 1989.

Girard 86 : J-Y. Girard, "Linear Logic", TCS vol.46, 1986.

Goltz : U. Goltz and A. Mycroft, "On the Relationship of CCS and Petri Nets", Lecture notes in Computer Science vol. 172, 1984.

Lamport 82 : L. Lamport, R. Shostak, and M. Pease, "The Byzantine Generals Problem", ACM Trans. on Prog. Lang. and Syst. vol.4, No.3, July 1982.

Milner 84 : R. Milner, "Calculi of Synchrony and Asynchrony", TCS, 1984.

Neilsen 79 : M. Neilsen, G. Plotkin and G. Winskel, "Petri Nets, Event Structures and Domains, Part 1", Internal Report CSR-47-79, November 1979.

Petri 76: C. A. Petri, "General net theory", *communication disciplines*, in B. Show Ed., Proc. Joint IBM University of Newcastle Seminar 1976.

Sun 87 : SUN, Yong, "Formal Specification of protocols and its independence of communication mechanisms", *communication in distributed systems*, Informatik Fachberichte, vol. 130, Springer-Verlag, 1987.

Winskel 83 : G. Winskel, "Event Structure Semantics for CCS and Related Languages", Lecture notes in Computer Science vol.140, 1983.

Winskel 87 : G. Winskel, "Petri Nets, Algebras, Morphisms, and Compositionality", Information and Computation (the old name was *information and control*) vol.72, 1987.