On equivalent execution semantics of concurrent systems

R. Janicki and M. Koutny

Abstract

Various execution semantics of concurrent systems are formally defined and investigated. The problem of the existence of minimal execution semantics equivalent to the semantics expressed intuitively as "execute as possible" is studied. The positive answer to that problem is given.


Series Editor: M.J. Elphick

© 1987 University of Newcastle upon Tyne.
Printed and published by the University of Newcastle upon Tyne,
Computing Laboratory, Claremont Tower, Claremont Road,
Newcastle upon Tyne, NE1 7RU, England.
Bibliographical details

JANICKI, Ryszard

On equivalent execution semantics of concurrent systems.


(University of Newcastle upon Tyne, Computing Laboratory, Technical Report Series, no. 234.)

Added entries

UNIVERSITY OF NEWCASTLE UPON TYNE.

Abstract

Various execution semantics of concurrent systems are formally defined and investigated. The problem of the existence of minimal execution semantics equivalent to the semantics expressed intuitively as "execute as possible" is studied. The positive answer to that problem is given.

About the author

Dr. R. Janicki visited the Computing Laboratory in 1982. From March 1984 until 1986 he was at the University of Aalborg, in Denmark. Then from March 1986 at the Department of Computer Science and Systems, McMaster University, Canada.

Dr. M. Koutny has been at the Computing Laboratory as a Research Associate from March, 1985 until Nov. 1986 and as a Lecturer from December, 1986. Prior to this he was at Warsaw Technical University in Poland from 1982.

Suggested keywords

ADEQUACY
CONCURRENCY
DEADLOCK-FREENESS
EQUIVALENCE
PETRI NETS
SEMANTICS

Suggested classmarks (primary classmark underlined)

Dewey (18th): 001.6424 001.64404
U.D.C. 519.682 681.322.06
ON EQUIVALENT EXECUTION SEMANTICS OF CONCURRENT SYSTEMS†

Ryszard Janicki
Department of Computer Science and Systems,
McMaster University,
1280 Main Street West, Hamilton, Ontario,
Canada L8S 4K1

Maciej Koutny
Computing Laboratory, The University of Newcastle upon Tyne,
Claremont Tower, Claremont Road,
Newcastle upon Tyne, NE1 7RU, U.K.

Abstract
Various execution semantics of concurrent systems are formally defined and investigated. The problem of the existence of minimal execution semantics equivalent to the semantics expressed intuitively as "execute as possible" is studied. The positive answer to that problem is given.

1 Introduction

In contrast to sequential systems for which there is only one execution semantics (each single execution is represented by a sequence of event occurrences), non-sequential systems may be described by means of various execution models. Perhaps the most common model is that of execution sequences. Informally, we can express this kind of semantics as: "execute as possible in sequence". Examples of such an approach, which directly follows the execution model successfully applied to sequential systems, include firing sequences of Petri nets ([BRA 80], [PET 81]), and interleaving semantics of CCS ([MIL 80]) and CSP ([HOA 80]). Another widely accepted approach, which is usually expressed as: "execute as possible (but not necessarily with maximal concurrency)", is that of step sequences. Here, computations are represented as sequences of sets

of events occurring concurrently. The step sequences semantics, and its distinguished variant, the maximally concurrent semantics, have been discussed for a number of models, c.f. [ELR 82], [JAN 86a,86b], [NIV 82], [REI 84], [ROZ 83], [SAL 81].

Despite twenty years of intensive research, in practice the analysis of the dynamic properties of real systems specified by Petri nets frequently forces us to analyze the reachability graph of the net. The reachability graph defined by the full (i.e. "execute as possible") semantics is typically very large and thus its analysis is very long and difficult task, even with the assistance of Computer-Aided Design systems (such as those in [JEN 83], [LAU 83] or [MON 83]). However, the reachability graph defined by the maximally concurrent semantics is much smaller than that defined by the full semantics. Consequently, systems for which the maximally concurrent behaviour determines the full behaviour are much easier to analyze. Unfortunately, the full and the maximally concurrent behaviour are not equivalent in the general case.

In this paper we try to overcome deficiencies of the maximally concurrent semantics by attempting to define an execution semantics which is equivalent to the full behaviour and has minimal reachability graph. We have concentrated here on equivalence with respect to two dynamic properties of concurrent systems, namely, deadlock-freeness and adequacy (a property similar to liveness which is akin to absence of partial system deadlock).

2 Preliminaries

The discussion presented in this paper is based on the notion of a Petri net decomposable into one-token finite state machines. In this section we introduce the basic notions and notation used throughout this paper, and formulate some useful facts. Note that since we limit our discussion only to Petri nets of a special structure the reader may find some of our notions as being restrictive, e.g., we do not consider nets with places having limited capacities, or weighted arcs.

Definition 2.1 A quadruple \( \Sigma = (S, T; F, M_0) \) is said to be a marked Petri net (shortly net) iff:

- \( S \) and \( T \) are finite non-empty sets;
- \( S \cap T = \emptyset \);
- \( F \subseteq (S \times T) \cup (T \times S) \);
- \( M_0 : S \to \mathbb{N} \);

where \( \mathbb{N} \) denotes the set of non-negative integers. Every mapping \( M : S \to \mathbb{N} \) is called a marking of \( \Sigma \), and for every \( x \in S \cup T \) we denote:

\[
*x = \{ y : (y,x) \in F \} \quad \text{and} \quad x^* = \{ y : (x,y) \in F \}.
\]

The elements of \( S \) and \( T \) are called places and transitions, respectively. \( F \) is the flow relation of \( \Sigma \), and \( M_0 \) is called the initial marking of \( \Sigma \). Diagrammatically we represent places and transitions as circles and boxes, respectively. The arcs joining boxes and circles indicate the flow relation, and any marking \( M \) is represented by drawing the appropriate number of black dots inside circles,
i.e. $M(s)$ dots inside the circle representing a place $s$. Figure 2.1 shows an example of a net $\Sigma = \{(1,2,3,4), \{a,b,c\}, \{1,a\}, (c,1), (2,b), (c,2), (3,c), (a,3), (b,3), (b,4)\}, M_0$ such that $M_0(1) = 0$, $M_0(2) = 1$, $M_0(3) = 2$ and $M_0(4) = 0$. We observe that $c = \{3\}$ and $c^* = \{1,2\}$.

We will now introduce the concept of step sequence semantics (see [ROZ 83]) in which a possible execution of a net is a step sequence. A step sequence is a string of single steps, each single step being a non-empty set of transitions firing simultaneously. Such an approach overcomes some of shortcomings of the execution sequences model in which a possible execution is a sequence of individual transition firings. For example, it distinguishes between arbitrary interleaving and concurrency.

![Figure 2.1: A net.](image)

**Definition 2.2** Let $\Sigma = (S, T; F, M_0)$ be a net, and let $M$ be its marking.

A non-empty set of transitions $A \subseteq T$ is $M$-enabled iff $M(s) \geq \text{card}(s^* \cap A)$ for every $s \in S$.

An $M$-enabled set $A$ is said to be a single step from marking $M$ to marking $M'$ such that:

$$\forall s \in S : M'(s) = M(s) - \text{card}(s^* \cap A) + \text{card}(s^* \cap A).$$

We denote this by $M[A > M'].$

**Definition 2.3** Let $\Sigma = (S, T; F, M_0)$ be a net. A step sequence of length $n$ ($n \geq 1$) of $\Sigma$ is defined to be any string $\sigma = A_1 \ldots A_n$ such that $A_1, \ldots, A_n \subseteq T$ and there exist markings $M_1, \ldots, M_n$ satisfying:

$$M_0[A_1 > M_1[A_2 > M_2 \ldots [A_n > M_n.$$  

We then denote $M_0[\sigma > M_n$. Also, we assume that the empty string is the empty step sequence of length zero. It will be denoted by $\lambda$, and we assume that $M_0[\lambda > M_0$.

Note that the length of a step sequence $\sigma$ will be denoted by $|\sigma|$. 

Referring to the example net of Figure 2.1 we have the following:

$$M_0\{(b,c) > M \text{ and } M\{(a,b,c) > M'\}.$$
where $M(1) = M(1) = 1, M(2) = M(2) = 1, M(3) = 2, M(3) = 3, M(4) = 1$ and $M(4) = 2$. Thus
$a = \{b, c\} \{a, b, c\}$ is a step sequence.

**Definition 2.4** A net $\Sigma = (S, T; F, M_0)$ is said to be a **one-token finite state machine** (shortly, state machine) iff:

- $\forall t \in T: \text{card}(t^*) = \text{card}(t^*) = 1$;
- $\forall s \in S: M_0(s) \leq 1$;
- $\text{card}(\{ s \in S : M_0(s) = 0 \}) = 1$. □

Intuitively, a state machine can be thought of as representing a sequential (sub-)system.

**Figure 2.2(a)** shows two examples of state machines.

![Diagram of state machines](image)

**Figure 2.2:** (a) Two state machines. (b) A net decomposable into these state machines.

**Definition 2.5** A net $\Sigma = (S, T; F, M_0)$ is said to be **decomposable into state machines** iff there are

$k \geq 1$ state machines $\Sigma_1, \ldots, \Sigma_k$ ($\Sigma_i = (S_i, T_i; F_i, M_i)$ for $i \leq k$) such that:

- $S = S_1 \cup \ldots \cup S_k$;
- $T = T_1 \cup \ldots \cup T_k$;
- $F = F_1 \cup \ldots \cup F_k$.
\[ \forall i \neq j \leq k : S_i \cap S_j = \emptyset ; \]
\[ \forall i \leq k \ \forall s \in S_i : M_o(s) = M_i(s). \]

The net of Figure 2.2(b) is decomposable into state machines \( \Sigma_1 \) and \( \Sigma_2 \) of Figure 2.2(a), whilst the net of Figure 2.1 cannot be decomposed into state machines. We note that the net \( \Sigma \) in the above definition can be regarded as a specification of a concurrent system which is composed of \( k \) sequential subsystems represented by the state machines \( \Sigma_1, \ldots, \Sigma_k \). The synchronization among those sequential subsystems is achieved through the sharing of transitions.

As we mentioned we will only discuss nets which are decomposable into state machines. We now \textbf{fix} such a net \( \Sigma = (S, T; F, M_o) \), and we \textbf{fix} \( k \geq 1 \) state machines \( \Sigma_i = (S_i, T_i; F_i, M_i) \) such that \( \Sigma \) can be decomposed into \( \Sigma_1, \ldots, \Sigma_k \).

Define:
\[ \text{ind} = \{(t, s) \in T \times T : \ \forall i \leq k : t \notin T_i \lor s \notin T_i \}; \]
and
\[ \text{Ind} = \{ A \subseteq T : A \neq \emptyset \land (\forall t \neq s \in A : (t, s) \notin \text{ind}) \}. \]

That is, \( \text{ind} \) is the independency relation on transitions such that two transitions are independent if they appear in different \( \Sigma_i \)'s. The set \( \text{Ind} \) comprises all (non-empty) sets of mutually independent transitions. For the net of Figure 2.2(b) we obtain:
\[ \text{ind} = \{(a, b), (b, a)\} \]
and
\[ \text{Ind} = \{(a), (b), (c), (d), (a, b)\}. \]

It turns out that the set of all step sequences of \( \Sigma \), denoted by \( SSEQ \), can be thought of as being a language over the alphabet \( \text{Ind} \).

**Lemma 2.6** \( SSEQ \subseteq \text{Ind}^* \).

**Proof (sketch)**

We observe that the thesis stems from the following property. If \( M_o (|o| > M \), where \( |o| \geq 0 \),
then one is the total number of dots inside the places of each net \( \Sigma_i \), which directly follows from the fact that \( M_o \) possesses this property and from \( \text{card}(t^* \cap S_i) = \text{card}(t' \cap S_i) = 1 \) for all \( t \in T_i \) and \( i \leq k \).

Below we introduce a number of notions that will be used in the next two sections.

For every alphabet \( \text{Alph} \) and every language \( L \subseteq \text{Alph}^* \), we use \( \text{Pref}(L) \) to denote the set of the prefixes of all strings in \( L \), i.e. \( \text{Pref}(L) = \{ x \in \text{Alph}^* : \exists y \in \text{Alph}^* : xy \in L \} \). If \( x \in \text{Alph}^* \) then we can write \( \text{Pref}(x) \) instead of \( \text{Pref}(xy) \).

If \( \sigma \in SSEQ \) then by denoting \( \sigma = A_1 \ldots A_n \) (or \( \sigma = B_1 \ldots B_m \)) we mean that \( A_i \in \text{Ind} \) (or \( B_j \in \text{Ind} \)) for all \( i \leq n \) (or \( j \leq m \)). Also, if \( n = 0 \) (or \( m = 0 \)) then \( \sigma = \lambda \).

Let \( \Gamma \subseteq SSEQ \) be a prefix-closed set, i.e. \( \text{Pref}(\Gamma) = \Gamma \). Then, for every \( \Omega \subseteq \Gamma \) we use \( \Gamma^\Omega \) to denote the postfix of \( \Omega \) in \( \Gamma \), i.e. we define
\[ \Gamma^\Omega = \{ \sigma \in \Gamma : \exists \omega \in \Omega : \omega \in \text{Pref}(\sigma) \}. \]

Also, if \( \sigma \in \Gamma \) then \( \Gamma^\sigma = \Gamma^{|\sigma|} \).

For example, if \( \Gamma = \{ \lambda, (a), (a)(b), (a)(c), (a)(c)(d), (a)(c)(e,f), (g), (g)(h) \} \), then \( \Gamma^\Omega = \{ (a)(c), \ldots \} \).
(a|c|d), (a|c|e|f), (g|h) and \( \Gamma = \{ (a), (a|b), \}, \), \( \Gamma = \{ (a|c), (a|c|d), (a|c|e), f \} \), for \( \Omega = \{ (a|c), (g|h) \} \) and \( \sigma = (a) \).

For every \( \Gamma \subseteq SSEQ \) and every \( \sigma \in \Gamma \) we denote \( \text{enabled}_\Gamma (\sigma) = \{ A \in \text{Ind} : \ oA \in \Gamma \} \). Also, we denote \( \text{maxenabled}(\sigma) = \{ A \in \text{enabled}_{\text{SSeq}}(\sigma) : \ \forall B \in \text{enabled}_{\text{SSeq}}(\sigma) : A \subseteq B \Rightarrow A = B \} \).

For the net of Figure 2.2(b) we obtain: \( \text{enabled}_{\text{SSeq}}(\lambda) = \{ (a), (a|b), \} \); \( \text{enabled}_{\text{SSeq}}(\{b\}) = \{ \lambda, (c) \}; \) \( \text{maxenabled}(\lambda) = \{ (a, b) \}; \) and \( \text{maxenabled}(\{b\}) = \text{enabled}_{\text{SSeq}}(\{b\}) \).

Note that \( \text{maxenabled}(\sigma) \) comprises maximal (in the sense of set inclusion ordering) steps that are \( M \)-enabled for the marking satisfying \( M_0 \sigma > M \).

Let for every \( i \leq k, \ |_i : \text{Ind}^* \rightarrow T^* \) be a homomorphism such that for every \( A \in \text{Ind} \),

\[
A|_i = \varepsilon \quad \text{if} \quad A \cap T_i = \emptyset \quad (\varepsilon \text{ denotes the empty string}); \\
A|_i = t \quad \text{if} \quad A \cap T_i = \{ t \}.
\]

Note that \( \mid_\cdot \) is a well-defined notion since \( \text{card}(A \cap T_i) \leq 1 \) for every \( A \in \text{Ind} \) and every \( i \leq k \).

The mappings \( \mid_1, \ldots, \mid_k \) play a central role in our discussion. As we mentioned, \( \Sigma \) represents a system composed of \( k \) subsystems represented by the \( \Sigma_i \)'s, and therefore each step sequence describes an execution of the composed system. There will be, however, situations where we directly refer to the executions of the subnets \( \Sigma_i \) being induced by a step sequence \( \sigma \in \text{SSeq} \), and we will use \( \sigma|_i \) to denote such an induced (or projected) execution.

Taking as an example the net of Figure 2.2(b) and \( \sigma = (a, b|d) \in \text{SSeq} \) we obtain: \( \sigma|_1 = ad \) and \( \sigma|_2 = bd \).

For every \( \Gamma \subseteq \text{SSeq} \) we denote

\[
[\Gamma] = \{ \sigma \in \text{SSeq} : \ \exists \tau \in \Gamma : \ \forall i \leq k : o_i = v_i \}; \\
\text{Seq}(\Gamma) = \{ \sigma \in \text{SSeq} : \ \exists \tau \in \Gamma : \ \forall i \leq k : o_i \in \text{Pref}(v_i) \}.
\]

If \( \sigma \in \Gamma \) then we can denote \( [\sigma] \) and \( \text{Seq}(\sigma) \) instead of \( [\{\sigma\}] \) and \( \text{Seq}(\{\sigma\}) \), respectively.

The set \([\Gamma]\) comprises all those executions of \( \Sigma \) which are equivalent with at least one step sequence of \( \Gamma \); two step sequences being equivalent if they induce the same sequential executions of the \( \Sigma_i \)'s. A step sequence belongs to \( \text{Seq}(\Gamma) \) if it can be derived from the set \([\Gamma]\) by using the prefix relation (see Lemma 2.7 below). For the net of Figure 2.2(b) and \( \Gamma = \{ (a|b) \} \) we obtain:

\[
[\Gamma] = \{ (a|b), (b|a), (a|b) \} \quad \text{and} \quad \text{Seq}(\Gamma) = \{ \lambda, (a), (b), (a|b), (b|a), (a|b) \}.
\]

**Lemma 2.7** If \( \tau \in \text{SSeq} \) and \( \sigma \in \text{Seq}(\tau) \) then there is \( \omega \in \text{SSeq} \) such that \( \sigma \in \text{Pref}(\omega) \) and \( \omega \in [\tau] \).  

One can easily see that if \( \sigma = \rho\text{AB} \) is a step sequence and \( t \in B \) is a transition which is independent with all transitions of the step A, then \( t \) can be transferred from \( B \) to \( A \) giving rise to a new step sequence. That is, we have the following.

**Lemma 2.8** If \( \sigma = \rho\text{AB} \in \text{SSeq} \), where \( A, B \in \text{Ind} \), and \( t \in B \) is such that \( \langle t, s \rangle \in \text{ind} \) for all \( s \in A \), then \( \tau = \rho (A \cup \{ t \}) \in \text{SSeq} \).  

-6-
We end this section introducing two important classes of step sequences.

**Definition 2.9** Let $\Delta$ denote the set of all step sequences $\sigma = A_1 \ldots A_n$ satisfying:

$$\forall 2 \leq j \leq n \ \forall t \in A_j \ \exists s \in A_{j-1} : \ (t, s) \notin \text{ind.}$$

Also, we use $\Xi$ to denote the set of all $\sigma = A_1 \ldots A_n \in \Delta$ such that either $n = 0$ (i.e. $\sigma = \lambda$), or $n \geq 1$ and $A_n \in \text{maxenabled}(A_1 \ldots A_{n-1})$. □

Note that if $\sigma \in SSEQ$ and $|\sigma| \leq 1$ then $\sigma \in \Delta$.

Intuitively, in any step sequence $\sigma = A_1 \ldots A_n \in \Delta$, all transitions can be thought of as being fired "as soon as possible" since no $t \in A_i$ can be transferred from $A_i$ to $A_{i-1}$. We also note that the set $\Xi$ contains those step sequences which are executed according to the "execute as much as possible" rule (see Lemma 3.3(a)).

Step sequences of $\Delta$ correspond to so-called interlace decompositions which have been introduced and investigated in [JAN 86b]. They also correspond to so-called normal form decomposition in the theory free partially commutative monoids ([Lal 79]). The lemma below directly follows from the results of [Lal 79, JAN 86b].

**Lemma 2.10**

(a) For every $\sigma \in SSEQ$, $\text{card}(\Delta \cap [\sigma]) = 1$.

(b) If $\sigma \in \Xi$ and $\tau \in \Delta$ satisfy $\sigma \in \text{Seq}(\tau)$ then $\sigma \in \text{Pref}(\tau)$. □

To illustrate the last two notions, we take once more take the net of Figure 2.2(b). Below we list all step sequences of $\Delta$ and $\Xi$ of length less or equal three.

$$\Delta = \{ \lambda, \{a\}, \{b\}, \{a,b\}, \{b,c\}, \{a,b\}/d, \{b,c\}/b, \{a,b\}/d/a, \{a,b\}/d/b, \{a,b\}/d/b, \{a,b\}/d/b, \{a,b\}/d/b, \ldots \};$$

$$\Xi = \{ \lambda, \{a,b\}, \{b,c\}, \{a,b\}/d, \{b,c\}/b, \{a,b\}/d/b, \{a,b\}/d/b, \ldots \}.$$

### 3 Execution Semantics

We have seen that the set of all step sequences $SSEQ$ can be interpreted as a language over the alphabet $\text{Ind}$ of all non-empty sets of mutually independent transitions; the independency relation $\text{ind}$ being determined by the distribution of the transitions over the $\Sigma_i$’s. On the other hand, $SSEQ$ provides a natural representation of the "execute as possible" semantics. These two facts suggest that the execution models of the net $\Sigma$ might be characterized by suitable lanuages over the alphabet $\text{Ind}$. We will call each such language a strategy. There are two properties that every strategy $\Gamma$ must satisfy, namely: (i) $\Gamma$ must be prefix-closed, and (ii) $\Gamma$ must be a subset of $SSEQ$. These two conditions are not, however, sufficient to rule out languages that can hardly be regarded as representing well-defined execution semantics. Consider, for example, the net of Figure 2.2(b) and the following set $\Gamma \subseteq \text{Ind}^*$:

$$\Gamma = \{ \lambda, \{a\}, \{a\}/b, \{a\}/d, \{a,b\} \}.$$

Clearly, $\Gamma$ satisfies (i) and (ii) but it cannot be accepted as a strategy. We explain this as follows. The step sequences $\sigma = \{a\}/b$ and $\tau = \{a,b\}$ are both executions of the same history of the system, so they should have the same continuations in $\Gamma$ (which is not the case). Our
argument is based on the principle that at any stage of an evolution the future behaviours (or future histories) of a concurrent system depend on its past history, but cannot depend on a particular way in which this history has been derived. We will therefore introduce a condition (iii) which says that if two step sequences of $\Gamma$ are equivalent (i.e. are both executions of the same system's history) then they have identical continuations in $\Gamma$. Out of many possible formalizations of the equivalence relation on step sequences we have chosen a somewhat "standard" one in which $\sigma$ and $\tau$ are equivalent, $\sigma \equiv \tau$, if they both induce the same histories of sequential subsystems represented by the $\Sigma_i$'s. The alternative is to say that $\sigma$ and $\tau$ are "linearizations" of the same underlying partial order of transition firings. The former interpretation is closely related to the standard semantics of the COSY path expression programs ([LAU 81], [SHI 79]); the latter is derived from the theory of traces of Mazurkiewicz ([MAZ 77]).

The three conditions (i), (ii) and (iii) are still not satisfactory. Let us again take the net of Figure 2.2(b) and the set $\Gamma \subseteq \text{Ind}^*$ such that:

$$\Gamma = \{\lambda, [a], \{a,b\}\}.$$ 

Clearly, $\Gamma$ satisfies (i), (ii) and (iii). Nevertheless the behaviour it specifies is rather difficult to interpret. For $\Gamma$ specifies that the system deadlocks after firing transition $a$, and despite this $\Gamma$ specifies that $a$ can be concurrently executed with transition $b$! We do not think that such a situation can actually happen in a concurrent system, and $\Gamma$ should not be regarded as a strategy. To exclude this kind of uninterpreted situations (and also others) we will introduce the fourth, and last, condition (iv) to be satisfied by any strategy $\Gamma$. In this condition we specify that if $\Omega \subseteq \Gamma$ is a complete distinguished part of the computation structure represented by $\Gamma$, then $\Omega$ must not be redundant, i.e. it cannot be deleted without losing the expressive power of $\Gamma$. More precisely, if a non-empty set $\Omega \subseteq \Gamma$ is such that: $\Omega$ is postifix-closed in $\Gamma$ and the set of histories generated by the step sequences of $\Omega$ is disjoint with the set of histories generated by the step sequences of $\Gamma - \Omega$, then there is $\omega \in \Omega$ such that $\omega$ cannot be derived from the step sequences generated by $\Gamma - \Omega$. It should be pointed out that in our approach a step sequence $\omega$ can be derived from a step sequence $\sigma$ if $\omega \in \text{Seq}(\sigma)$.

We may check that $\Gamma = \{\lambda, [a], \{a,b\}\}$ does not satisfy (iv). Indeed, $\Omega = \{[a]\} \subseteq \Gamma$ is postifix-closed in $\Gamma$ and the sets of histories generated by $\Omega$ and $\Gamma - \Omega = \{\lambda, \{a,b\}\}$ are disjoint, but $\omega = \{a\}$ can be derived from $\sigma = \{a,b\} \in \Gamma - \Omega$ since: $\tau = \{a\}/b$ and $\sigma$ are equivalent, and $\omega$ is a prefix of $\tau$. Formally, we proceed as follows.

Let $\equiv$ be a binary relation on step sequences such that for all $\sigma, \tau \in SSEQ$,

$$\sigma \equiv \tau \iff (\forall i \leq k : o_i = v_i).$$

Clearly, $\equiv$ is an equivalence relation, and its set of equivalence classes will be denoted by $Hist$. The relation $\equiv$ should be understood as follows: if $\sigma \equiv \tau$ then the step sequences $\sigma$ and $\tau$ are both executions of the same system's history which is itself represented by the equivalence class (history) $h \in Hist$ such that $\sigma, \tau \in h$. (Note that $h = [\sigma] = [\tau]$.)
Definition 3.1 A non-empty language $\Gamma \subseteq \text{Ind}^*$ is said to be a strategy iff:

$$\Gamma \subseteq \text{SSEQ}; \tag{3.1}$$

$$\text{Pref}(\Gamma) = \Gamma; \tag{3.2}$$

$$\forall \sigma, \tau \in \Gamma : \sigma \approx \tau \Rightarrow \text{enabled}_\Gamma(\sigma) = \text{enabled}_\Gamma(\tau); \tag{3.3}$$

$$[\Omega] \cap [\Gamma - \Omega] = \varnothing \Rightarrow \Omega \cdot \text{Seq}(\Gamma - \Omega) = \varnothing \tag{3.4}$$

where (3.4) is satisfied for every non-empty set $\Omega \subseteq \Gamma$ such that $\Omega = \Gamma \Omega$. □

Note that in (3.4) $\Omega$ is a set which is postfixed in $\Gamma$, and such that $\neg (\sigma \approx \tau)$ for every $\sigma \in \Omega$ and every $\tau \in \Gamma - \Omega$. That is, $\Omega$ is a complete distinguished part of the computation structure represented by $\Gamma$.

Lemma 3.2 $\text{SSEQ}$ is a strategy.

Proof

Clearly, $\text{SSEQ} \neq \varnothing$, and (3.1) and (3.2) are satisfied.

To prove (3.3) we take $\sigma, \tau \in \text{SSEQ}$ such that $\sigma \approx \tau$ and $A \in \text{enabled}_{\text{SSEQ}}(\sigma)$.

We have $M_\sigma[A > M_1 \ A > M_2$ and $M_\sigma[A \tau > M_3$. Since $|\sigma| = |\tau| \Rightarrow \text{true}$ for all $i \leq k$, $M_1 = M_3$.

Thus $A \in \text{enabled}_{\text{SSEQ}}(\tau)$.

Consider now $\Omega \subseteq \text{SSEQ}$ satisfying: $[\Omega] \cap [\text{SSEQ} \cdot \Omega] = \varnothing$ and $\Omega = \text{SSEQ} \Omega \neq \varnothing$.

Let $\sigma \in \Omega$ and let $\tau \in \text{SSEQ}$ be such that $\sigma \in \text{Seq}(\tau)$. By Lemma 2.7, there is $\omega$ such that $\tau = \omega$ and $\sigma \in \text{Pref}(\omega)$. Thus, by $\Omega = \text{SSEQ} \Omega$, $\omega \in \Omega$. Hence, by $[\Omega] \cap [\text{SSEQ} \cdot \Omega] = \varnothing$ and $\tau = \omega$, $\tau \in \Omega$. Consequently, $\Omega - \text{Seq}(\text{SSEQ} \cdot \Omega) = \Omega \neq \varnothing$, which completes the proof of (3.4).

The maximally concurrent semantics (expressed informally as "execute as much as possible in parallel") can be represented by a language $\text{MAX}$ of those step sequences whose single steps are maximal in the sense that no additional independent transitions can be added.

We define $\text{MAX}$ as the least set of step sequences such that:

$$\lambda \in \text{MAX}; \quad \forall \sigma \in \text{MAX} \quad \forall A \in \text{maxenabled}(\sigma) : \ oA \in \text{MAX}.$$ 

For the net of Figure 2.2(b) we obtain:

$$\text{MAX} = \{ \lambda, \{a,b\}, \{a,b\}[d], \{a,b\}[d][a,b], \{a,b\}[d][a,b][d], \{a,b\}[d][a,b][d][a,b], ... \}.$$ 

It turns out that $\text{MAX}$ is a strategy whose elements are step sequences of the set $\Xi$ that has been introduced in Definition 2.9.

Lemma 3.3

(a) $\text{MAX} \subseteq \Xi$;

(b) $\text{MAX}$ is a strategy.

Proof

(a) We will prove that $\sigma \in \text{MAX}$ implies $\sigma \in \Xi$ by induction on the length of $\sigma$.

Clearly, if $|\sigma| = 1$ then $\sigma \in \Xi$.

Assume that the thesis holds for all $\sigma \in \text{MAX}$ such that $|\sigma| = n \geq 1$. 

-9-
Let $\sigma = A_1...A_nA_{n+1} \in MAX$. Suppose that $\sigma \not\in \Xi$. By the induction hypothesis, $\tau = A_1...A_n \in \Xi$. Hence, since $A_{n+1} \in maxenabled(\tau)$, there must be $t \in A_{n+1}$ such that $(t, \sigma) \in ind$ for all $s \in A_n$. Consequently, by Lemma 2.8, $\omega = A_1...A_n(A_n \cup \{t\}) \in SSEQ$. Thus $A_n \in maxenabled(A_1...A_n, 1)$, a contradiction.

Hence $\sigma \in \Xi$, which completes the proof.

(b) Clearly, $MAX = \varnothing$, and (3.1) and (3.2) are satisfied. We also note that if $\sigma = \tau \in MAX$ then, by $\sigma, \tau \in \Xi \subseteq \Delta$ and Lemma 2.10(a), $\neg (\sigma = \tau)$. Hence (3.3) is also true.

Let $\Omega \subseteq MAX$ be such that $\Omega = MAX^\ominus$ and $\Omega \neq \varnothing$. Suppose that $\sigma \in \Omega$ and $\tau \in MAX$ satisfy $\sigma \in Seq(\tau)$. Then, by Lemma 2.10(b) and (a), $\sigma \in Pref(\tau)$. Thus, by $\Omega = MAX^\ominus$, $\tau \in \Omega$.

That is, we have shown that $\Omega \setminus Seq(MAX - \Omega) = \varnothing$, which completes the proof of (3.4).

The discussion which follows will focus on semantical equivalence between strategies with respect to deadlock-freeness and adequacy. Both properties can be easily expressed in terms of the general execution semantics.

**Definition 3.4** Let $\Gamma$ be a strategy.

- $\Sigma$ is $\Gamma$-deadlock-free iff: $\forall \sigma \in \Gamma: enabled_\Gamma(\sigma) \neq \varnothing$.
- $\Sigma$ is $\Gamma$-adequate iff: $\forall \sigma \in \Gamma \; \forall t \in T \; \exists \tau \in \Gamma^0 \; \exists A \in enabled_\Gamma(\tau): t \in A$. □

The $SSEQ$-deadlock-freeness and $SSEQ$-adequacy will be called deadlock-freeness and adequacy, respectively.

The $\Gamma$-deadlock-freeness means that each computation in $\Gamma$ can be continued, while the $\Gamma$-adequacy says that no computation in $\Gamma$ leads to a situation which could prevent any transition from being fired in the future.

In [JAN 83] and [JAN 86b] the relationships between deadlock-freeness (adequacy) and $MAX$-deadlock-freeness ($MAX$-adequacy) were discussed in the context of the COSY path expression formalism. The theorem below is a generalization of a result obtained there whose essential meaning was that $SSEQ$ and $MAX$ are behaviourally equivalent provided that $SSEQ = Seq(MAX)$. (Note that this is not satisfied for the net of Figure 2.2(b), e.g. $\sigma = \{b\}(c) \not\in Seq(MAX)$.) The theorem below provides us with a similar simple sufficient condition for behavioural equivalence of two arbitrary execution semantics.

**Theorem 3.5** If $\Psi$ and $\Gamma$ are two strategies satisfying $Seq(\Psi) = Seq(\Gamma)$ then:

(a) $\Sigma$ is $\Psi$-deadlock-free $\iff \Sigma$ is $\Gamma$-deadlock-free

(b) $\Sigma$ is $\Psi$-adequate $\iff \Sigma$ is $\Gamma$-adequate. □

It is worth to emphasize an intuitive meaning of the condition $Seq(\Psi) = Seq(\Gamma)$. For a given strategy $\Phi$, the set $Seq(\Phi)$ comprises all possible execution paths that can be derived from the step sequences of $\Phi$, and thus $Seq(\Phi)$ may be regarded as a characterization of the expressive
power of $\Phi$. Consequently, $\text{Seq}(\Psi) = \text{Seq}(\Gamma)$ means that the strategies $\Psi$ and $\Gamma$ have the same expressive power, or that they convey the same information.

**Proof of Theorem 3.5**

(We only prove the $\Rightarrow$ implications)

(a) Suppose that $\Sigma$ is $\Psi$-deadlock-free, but is not $\Gamma$-deadlock-free.

Let $\sigma \in \Gamma$ be a step sequence with $\text{enabled}_\Gamma(\sigma) = \emptyset$. By $\text{Seq}(\Psi) = \text{Seq}(\Gamma)$, there is $\tau \in \Psi$ such that $\sigma \in \text{Seq}(\tau)$. Clearly, since $\Sigma$ is $\Psi$-deadlock-free, $\tau A \in \Psi$ for some $A \in \text{Ind}$. Hence, again by $\text{Seq}(\Psi) = \text{Seq}(\Gamma)$, there is $\omega \in \Gamma$ such that $\tau A \in \text{Seq}(\omega)$. Thus $\sigma \in \text{Seq}(\omega)$ and, of course, $\neg \sigma = \omega$.

Let $\Omega = \Gamma \cap [\omega]$. By (3.3) and $\text{enabled}_\Gamma(\sigma) = \emptyset$, $\Gamma^\Omega = \Omega = \emptyset$. Moreover, $[\Omega] \cap [\Gamma \setminus \Omega] = \emptyset$. Clearly, we have $\Omega \subseteq \text{Seq}(\omega)$. On the other hand, $\omega \in \Gamma \setminus \Omega$ (since $\neg \sigma = \omega$). Hence $\Omega \subseteq \text{Seq}(\Gamma \setminus \Omega) = \emptyset$, contradicting (3.4). This completes the proof of (a).

(b) Suppose that $\Sigma$ is $\Psi$-adequate, but is not $\Gamma$-adequate.

Let $\sigma \in \Gamma$ and $t \in T$ be such that for every $\tau \in \Gamma^\omega$ and every $A \in \text{enabled}_\tau(\sigma)$, $t \not\in A$. We first prove the following:

$$\forall \zeta \in \Gamma^\omega \exists \phi \in \Gamma: \zeta \in \text{Seq}(\phi) \wedge \phi \in \text{Seq}(\Gamma^\omega)$$

(3.5)

Let $\zeta \in \Gamma^\omega$. By $\text{Seq}(\Psi) = \text{Seq}(\Gamma)$, there is $\tau \in \Psi$ such that $\zeta \in \text{Seq}(\tau)$. Hence, since $\Sigma$ is $\Psi$-adequate, we have $\tau A \in \Psi$ and $t \in A$, for some $\rho \in \text{Ind}$ and $A \in \text{Ind}$. Thus, again by $\text{Seq}(\Psi) = \text{Seq}(\Gamma)$, there is $\phi \in \Gamma$ such that $\tau A \in \text{Seq}(\phi)$. Clearly, $\zeta \in \text{Seq}(\phi)$.

Let $l \leq k$ be such that $t \in T_l$, and let $t_\omega$ denotes the number of occurrences of $t$ within $\omega|_l$, for every $\omega \in \text{SSEQ}$. We observe that $t_\phi \geq t_\omega + 1$ and $t_\omega = t_\phi$ for every $\omega \in \Gamma^\omega$. Consequently, for every $\omega \in \Gamma^\omega$, $\phi|_l \in \text{Pref}(\omega|_l)$, so $\phi \in \text{Seq}(\omega)$. Hence (3.5) is satisfied.

Let $\Omega = \Gamma \cap [\Gamma^\omega]$. From (3.3) it follows that $\Gamma^\Omega = \Omega = \emptyset$. Moreover, $[\Omega] \cap [\Gamma \setminus \Omega] = \emptyset$.

Suppose that $\omega \in \Omega$. Then there is $\zeta \in \Gamma^\omega$ such that $\omega = \zeta$. Hence, by (3.5) there is $\phi \in \Gamma$ such that $\zeta \in \text{Seq}(\phi)$ and $\phi \in \text{Seq}(\Gamma^\omega)$ which means that $\phi \in \Gamma \setminus \Omega$. Thus we obtain $\omega \in \text{Seq}(\phi)$ and $\phi \in \Gamma \setminus \Omega$. Consequently, $\Omega \setminus \text{Seq}(\Gamma \setminus \Omega) = \emptyset$, a contradiction with (3.4). $\Box$

4 Minimal Execution Semantics

As it was shown in [JAN83, JAN86b], deadlock-freeness and adequacy expressed in terms of the SSEQ and MAX strategies are in the general case non-equivalent properties. This leaves open a question whether there exists an efficient strategy useful in determining the deadlock-freeness and adequacy of $\Sigma$, where by an efficient strategy we mean one that generates "minimal" reachability graph. In this section we will demonstrate that such a strategy can be found.

Intuitively, the strategy we are looking for should avoid unnecessary elaboration of execution paths. Also, we will demand the satisfaction of the precondition of Theorem 3.5.
Definition 4.1 A strategy $\Gamma$ is said to be solid iff $\text{Seq}(\Gamma) = \text{Seq}(\text{SE}_{Q}) = S_{Q}$. A solid strategy $\Gamma$ is said to be minimal iff it does not contain any proper solid substrategy, i.e. if $\Lambda \subseteq \Gamma$ is a solid strategy then $\Lambda = \Gamma$. □

After restricting our search only to minimal solid strategies we may look closer at the reachability graphs they generate. Our objective will be to distinguish those strategies which correspond to an intuitive meaning of efficient execution semantics.

Let us take as an example the net $\Sigma$ of Figure 4.1(a). One can easily see that the following three strategies are solid and minimal:

$\Gamma = \{ \lambda, \{a\}, \{a\}{(b)} \}$;
$\Psi = \{ \lambda, \{b\}, \{b\}{(a)} \}$;
$\Phi = \{ \lambda, \{a,b\} \}$.

The reachability graphs of $\Gamma$, $\Psi$ and $\Phi$ are shown in Figure 4.1(b).

Looking at the graphs of Figure 4.1(b) we may observe that the deadlock situation is detected by $\Phi$ in one move, whilst $\Gamma$ or $\Psi$ need both two moves. Such a property distinguishes $\Phi$ as a strategy being "faster" than $\Gamma$ or $\Psi$.

We therefore conclude that a strategy we are looking for should be fast. More specifically, for every history of the system there should be at least one execution path generating this history in possibly the smallest number of steps. To formulate this more precisely we need an auxiliary notion.

For every solid strategy $\Gamma$ and for every $h \in \text{Hist}$, we define $\text{min}_{\Gamma}(h) = \min \{ |o| : o \in \Gamma \land h \subseteq \text{Seq}(o) \}$.

Note that $\text{min}_{\Gamma}(h)$ denotes the length of a minimal step sequence of $\Gamma$ from which the step sequences of $h$ may be derived.
Definition 4.2 If $\Gamma$ and $\Psi$ are solid strategies then $\Gamma$ is said to be not slower than $\Psi$ iff $\min_{\Gamma}(h) \leq \min_{\Psi}(h)$ for all $h \in \text{Hist}$. □

For the strategies of Figure 4.1 we obtain the following:

<table>
<thead>
<tr>
<th></th>
<th>$h_0$</th>
<th>$h_1$</th>
<th>$h_2$</th>
<th>$h_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\min_{\Gamma}$</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>$\min_{\Psi}$</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>$\min_{\Phi}$</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

where $h_0 = \langle \lambda \rangle$, $h_1 = \{a\}$, $h_2 = \{b\}$ and $h_3 = \{a,b\}$. Thus, $\Phi$ is not slower than $\Gamma$ and $\Psi$, but not vice versa. We also observe that $\Gamma$ and $\Psi$ are incomparable in the sense of Definition 4.2.

Our problem now is whether there is a minimal solid strategy which is not slower than $\text{SSEQ}$. Below we prove that $D = \text{Pref}(\Xi)$, is such a strategy.

One can easily see that for the net of Figure 2.2(b) the following are the step sequences of length less or equal three belonging to $D$:

$$D = \{\lambda, \{a,b\}, \{a,b\}_{\{d\}}, \{a,b\}_{\{d\}{\{a,b\}}}, \{b\}, \{b\}{\{c\}}, \{b\}{\{c\}{\{b\}}}, \ldots \}.$$ 

Theorem 4.3 $D = \text{Pref}(\Xi)$ is a minimal solid strategy that is not slower than $\text{SSEQ}$.

Proof

For every $h \in \text{Hist}$ let $h_\Delta$ denote the step sequence such that $h_\Delta \in (h \cap \Delta)$. By Lemma 2.10(a), $h_\Delta$ is a well-defined notion.

For every $a \in \text{SSEQ}$ let $#(a)$ denote the total number of transition within $a$, i.e. $#(\lambda) = 0$, and if $a = A_1 \ldots A_n$, where $n \geq 1$, then $#(a) = \text{card}(A_1) + \ldots + \text{card}(A_n)$.

We first prove the following fact:

$$\forall h \in \text{Hist} \exists \sigma \in \Xi : h \subseteq \text{Seq}(a) \land |h_\Delta| \geq |\sigma|. \quad (4.1)$$

(That is, for every history $h$ there is a step sequence of $\Xi$ which generates $h$ in no more steps than $|h_\Delta|$.)

Suppose that (4.1) is not satisfied for some $h \in \text{Hist}$. Define $F = \{f \in \text{Hist} : |h_\Delta| \geq |f_\Delta| \land h \subseteq \text{Seq}(f)\}$. Clearly, $F$ is finite (since $|h_\Delta| \geq |f_\Delta|$ for all $f \in F$) and non-empty (since $h \in F$). Thus there is $g \in F$ such that $|(g_\Delta)| \geq |(f_\Delta)|$ for all $f \in F$.

Let $g_\Delta = A_1 \ldots A_n$. If $n = 0$ then (4.1) is satisfied, so we assume $n \geq 1$.

Since $h$ does not satisfy (4.1), $g_\Delta \in \Xi$, so there is $t \in A_n$ such that $t = A_1 \ldots A_{n-1}(A_n \cup \{t\}) \in \text{SSEQ}$. Let $e \in \text{Hist}$ be such that $e \cap t \notin e$. One may easily see that $e \in F$. Hence, $|(g_\Delta)| \geq |(e_\Delta)|$, contradicting $|(e_\Delta)| = 1 + |(g_\Delta)|$. Thus (4.1) is satisfied.

We can now proceed with the main proof. First, we will show that $D$ is a strategy.

One can easily see that $D$ satisfies: $D \neq \emptyset$, (3.1), (3.2) and (3.3). To show (3.4) we take a non-empty set $\Omega \subseteq D$ such that $\Omega = D^\Omega$. Then, by $D = \text{Pref}(\Xi)$, $\Xi \cap \Omega = \emptyset$. Let $a \in \Xi \cap \Omega$. Suppose that $t \in D$ satisfies $a \in \text{Seq}(t)$. Then, by Lemma 2.10(b), $a \in \text{Pref}(t)$. Thus, by $\Omega = D^\Omega$, we have $t \in \Omega$, so $a \in (\Omega - \text{Seq}(D - \Omega))$. Consequently, (3.4) is satisfied.
Thus $D$ is a strategy, and from (4.1) it follows that $D$ is solid.

Suppose that $\Gamma \subseteq D$ is a solid strategy. Then, since $D = \text{Pref}(\Xi)$, there is $\sigma \in \Xi$ such that $\sigma \notin \Gamma$. By $\text{Seq}(\Gamma) = \text{SSEQ}$, there is $\tau \in \Gamma$ such that $\sigma \in \text{Seq}(\tau)$. Hence, by Lemma 2.10(b), $\sigma \in \text{Pref}(\tau)$. Thus $\sigma \in \Gamma$, a contradiction. Consequently, $D$ is a minimal solid strategy.

Suppose now that $h \in \text{Hist}$. If $|h_\Delta| \leq 1$ then $\min_{\text{SSEQ}}(h) = \min_D(h)$ is trivially satisfied.

Assume that $h_\Delta = A_1 \ldots A_n$, where $n \geq 2$. By (4.1) there is $\omega \in \Xi$ such that $h \subseteq \text{Seq}(\omega)$ and $|\omega| \leq n$. Hence $\min_D(h) \leq n$. On the other hand, we observe that from the definition of $\Delta$ it follows that there are $t_i \in A_i$ (for $i = 1, \ldots, n$) such that $(t_i, t_{i+1}) \notin \text{ind}$ for $i = n, n-1, \ldots, 2$.

Suppose that $\sigma = B_1 \ldots B_m$ is such that $h_\Delta \subseteq \text{Seq}(\sigma)$. By $h_{\Delta j} \in \text{Pref}(\sigma_j)$ for all $j \leq k$ and $(t_i, t_{i+1}) \notin \text{ind}$ (for $i = n, n-1, \ldots, 2$) we have the following: for every $i = 1, \ldots, n$ there is $l_i \leq m$ such that $t_i \in B_{l_i}$, and $i \neq j \Rightarrow l_i \neq l_j$. Thus $m \geq n$ which means that $n \leq \min_{\text{SSEQ}}(h)$.

Thus, $\min_D(h) \leq n \leq \min_{\text{SSEQ}}(h)$ which completes the proof. □

![Diagram](image)

**Figure 4.2:** Minimal solid strategies not slower than SSEQ.

In general, there may be more than one minimal solid strategy which is not slower than SSEQ. Consider, for example, the net of Figure 4.2(a) and the following two minimal solid strategies $\Gamma$ and $\Psi (= D)$:

$\Gamma = \{ \lambda, \{a\}, \{a\}[d,b], \{a,b,c\} \}$

$\Psi = \{ \lambda, \{a,b\}, \{a,b\}[d], \{a,b,c\} \}$

The reachability graphs of $\Gamma$ and $\Psi$ are shown in Figure 4.2(b). One may easily see that $\Gamma$ and $\Psi$ are both not slower than SSEQ.

Finally, we would like to make an important remark. From $\text{MAX} \subseteq \Xi$ and $D = \text{Pref}(\Xi)$ we have $\text{MAX} \subseteq D$. Thus, if $\text{MAX}$ is solid then $D = \text{MAX}$ due to the minimality of $D$. We can therefore regard $D$ as being a natural generalization of the maximally concurrent semantics.
5 Concluding Remarks

The results presented show that it is always possible to execute a concurrent system represented by a Petri net decomposable into one-token finite state machines in such a way that one can reason about deadlock-freeness and adequacy, and in doing so to generate a minimal reachability graph. Although we have concentrated on only two properties of concurrent systems, i.e. deadlock-freeness and adequacy, we envisage the possibility of similarly treating other properties, e.g. fairness.

6 Acknowledgement

The authors are grateful to the referees for their valuable comments and suggestions.

The work of the second author was supported by a Grant from the Science and Engineering Council of Great Britain.

7 References


