ON THE APPLICATION OF EXTENDED PETRI NETS
to the VERIFICATION OF PROTOCOLS

Amir Tomer* and Michael Yoeli+

Technical Report # 248
November 1981
Revised Version  June 1982

*This paper is part of the first author's M.Sc. Thesis, Dept. of
Computer Science, Technion.

+This research was supported in part by the Fund for the Promotion
of Research at the Technion, Grant No. 120-589.

This paper was presented at the "Second European Workshop on Application
and Theory of Petri Nets"; Bad Honnef, West Germany, Sept. 28-30, 1981.
ABSTRACT

In this paper we apply the concept of Extended Petri Net, in order to develop a relatively simple, mathematically precise model of the Alternating Bit Protocol, including its capability to recover from failures. We demonstrate that this model is powerful enough to prove total correctness. For this purpose, we introduce a "Principle of Fairness", as abstraction of the idea that no total breakdown may occur.
1. INTRODUCTION

Considerable interest presently exists in a methodology for proving communication protocols correct [SUN-79], [MER]. The Alternating-Bit Protocol [BA-SC-WI] frequently serves as convenient example by means of which proposed methods can be easily illustrated. Important contributions to the modelling of the Alternating-Bit Protocol (ABP) are [BO-GE], [BR-DR], [SUN 80], [GIR], [HA-OW], [PO-FA]. As to [BO-GE], they do not provide a precise, mathematical formalism, but rely to a considerable extent on intuitive arguments, particularly with respect to "distantly initiated actions". Also, their correctness proof is based on the restricting assumption that a time-out transition will only occur after a transmission loss has occurred. This restricting assumption, which has no justification in real-life systems, has indeed been eliminated in [GIR]. The formally precise model of [GIR] covers a wide range of possible failures. However, both the model and the correctness proof in [GIR] are of considerable complexity. In this
paper we apply the concept of Extended Petri Net [YOE], in order to develop a relatively simple, mathematically-precise model of the ABP, including its capability to recover from failures. We demonstrate that this model is powerful enough to prove total correctness of the ABP. Similarly to [GIR], our proof does not rely on the above restricting assumption of [BO-GE].

Further references to related works on protocols are given in the last section, after our approach and model have been presented.

2. INFORMAL DESCRIPTION OF THE ABP

2.1 The Fault-Free ABP

The ABP is a communication protocol between two parties - SENDER and RECEIVER (Fig. 1). The SENDER produces a sequence of data messages, labeled alternatively by the control bits Q and 1. Each message is transmitted to the RECEIVER, which uses ("consumes") it and then returns an acknowledge message, labeled correspondingly. If we assume the performance of the ABP to be completely fault-free, Fig. 1 (cf. [BO-GE]) may be considered a basic model representing the ABP as two cooperating finite-state machines. The model of Fig. 1 is easily converted into a marked graph (cf. [MER]). Hence the theory of marked graphs is immediately applicable, to show that the system is live and operates correctly.

2.2 The ABP with Fault-Recovery

The problem of modeling the ABP becomes more difficult, if we also wish to incorporate the capabilities of the protocol to recover from faults (cf. [BA-SC-WI], [BO-GE], [BR-DR], [GIR]). Particularly, we are concerned with the following types of possible faults:
Legend

PROD = Produce new data message
SD\textsubscript{i} = Send data message labeled i \in \{0; 1\}
RD\textsubscript{i} = Receive data message labeled i
SA\textsubscript{i} = Send acknowledge message labeled i
RA\textsubscript{i} = Receive acknowledge message labeled i
CONS = Use (consume) message received

\textbullet{} = Initial state

FIGURE 1 - MODEL OF FAULT-FREE ABP
(a) loss of messages in either direction,
(b) delay of messages, even beyond any given time-out limit,
(c) errors in data messages, detected by the RECEIVER,
(d) errors in acknowledge messages, detected by the SENDER.

In the next section we introduce a suitable model, which also takes into account the fault-recovery features of the ABP.

3. A FORMAL MODEL OF THE ABP

3.1 Definition of Extended Petri Net (EPN)

Our formal model of the ABP is based on the concept of EPN [YOE 80]. The EPN is a slight modification of the concept of "parallel program" as defined in [KEL]. We denote by \( \omega \) the set of non-negative integers.

**Definition 3.1**

An Extended Petri Net (EPN) is an 8-tuple

\[ E = (P, T, F, M_0, D, C, A, d_o) \]

where

- **P** is a finite set of places;
- **T** is a finite set of transitions; \( T \cap P = \emptyset, T \cup P \neq \emptyset; \)
- **F** = \( (P \times T) \cup (T \times P) \) is the flow relation;
- **M_0** : \( P \rightarrow \omega \) is the initial marking;
- **D** is a set of data;
- **C** : \( T \times D \rightarrow \{ \text{true, false} \} \) is the condition function;
- **A** : \( T \times D \rightarrow D \) is the action function;
- \( d_o \in D \) represents the initial data.

A state of **E** is an ordered pair \((M,d)\), where \( M : P \rightarrow \omega \) and \( d \in D \).

The transition \( t \in T \) is enabled in state \((M,d)\) in symbols: \( (M,d)[t>\), iff

1. \( M[t], \) i.e. \( (\forall p \in P) [pFt \Rightarrow M(p) \geq 0] \)
2. \( C(t,d) = \text{true} \)
For states \((M, d), (M', d'), \) and \(t \in T\) we set

\[(M, d)[t > (M', d')] \iff (1) \quad (M, d)[t >
\]

\[(2) \quad M[t > M', \text{ i.e.}
\]

\[(\forall p \in P)[M'(p) = M(p) - W(p, t) + W(t, p)]
\]

where \(W(x, y) = \begin{cases} 1 & \text{if } x < y \\ 0 & \text{else} \end{cases}\)

\[(3) \quad d' = A(t, d),
\]

The above notation is extended, as usual, to finite strings of transitions. Namely, let

\[(M_i, d_i)[t_i > (M_{i+1}, d_{i+1}) \quad \text{for } 1 \leq i \leq k.
\]

In this case we set

\[(M_1, d_1)[t_1 t_2 \ldots t_k >
\]

as well as

\[(M_1, d_1)[t_1 t_2 \ldots t_k > (M_{k+1}, d_{k+1}).
\]

Furthermore, for an infinite string of transitions, \(\alpha = t_1 t_2 \ldots t_i \ldots\)

we set \((M_1, d_1)[\alpha >
\]

iff there exist states \((M_i, d_i), i = 2, 3, \ldots\), such that

\[(M_i, d_i)[t_i > (M_{i+1}, d_{i+1}), i = 1, 2, 3, \ldots.
\]

In the graphical representation of an EPN, places are represented by circles, transitions by bars, and the flow relation by directed arcs, as usual for Petri nets. Any transition \(t\) is labelled by \([\text{cond}(d)/d + f(d)]\) where 'cond' represents the unary predicate on \(D\), defined by

\[(\forall d \in D) \ [\text{cond}(d) \Rightarrow C(t, d)] ;
\]

\[f: D \rightarrow D \text{ is a function defined by}
\]

\[(\forall d \in D) \ [f(d) = A(t, d)].\]
We omit the first part of the label, if \( \text{cond}(d) \) holds for every \( d \in D \); the second part is omitted, if \( f \) is the identity function.

3.2 Definition of ABP-System

We shall now apply the above definition of EPN, in order to formally introduce the concept of an ABP-system. For a finite alphabet \( \Sigma \), we denote by \( \Sigma^\omega \) the set of all infinite strings over \( \Sigma \).

**Definition 3.2**

An **ABP-System** is an EPN \( E_{\text{ABP}} \) represented graphically in Fig. 2. The domains, initial values, and interpretations of the variables appearing in Fig. 2 are listed in the following Table 1.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Domain</th>
<th>Initial Value</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>SB</td>
<td>( {0,1} )</td>
<td>1</td>
<td>control bit attached to data message sent</td>
</tr>
<tr>
<td>SA</td>
<td>( {0,1} )</td>
<td>1</td>
<td>control bit attached to acknowledge message sent</td>
</tr>
<tr>
<td>RB</td>
<td>( {0,1} )</td>
<td>1</td>
<td>control bit attached to data message received</td>
</tr>
<tr>
<td>RA</td>
<td>( {0,1} )</td>
<td>1</td>
<td>control bit attached to acknowledge message received</td>
</tr>
<tr>
<td>BL</td>
<td>( {0,1} )</td>
<td>1</td>
<td>link for transmission of control bit attached to data message</td>
</tr>
<tr>
<td>AL</td>
<td>( {0,1} )</td>
<td>1</td>
<td>link for transmission of control bit attached to acknowledge message</td>
</tr>
<tr>
<td>ML</td>
<td>( \Sigma )</td>
<td>( \lambda )</td>
<td>content of message link</td>
</tr>
<tr>
<td>RM</td>
<td>( \Sigma )</td>
<td>( \lambda )</td>
<td>message received</td>
</tr>
<tr>
<td>PLIST</td>
<td>( \Sigma^\omega )</td>
<td>PLIST</td>
<td>constant infinite list of messages (Producer List)</td>
</tr>
<tr>
<td>CLIST</td>
<td>( \Sigma^* )</td>
<td>( \lambda )</td>
<td>list of messages received (Consumer List)</td>
</tr>
<tr>
<td>PT</td>
<td>( \omega )</td>
<td>0</td>
<td>pointer for PLIST</td>
</tr>
</tbody>
</table>

**TABLE 1 - DATA SET OF ABP-SYSTEM**
Figure 2: Graph-Representation of $E_{ABP}$.

Symbols: $\oplus$ - addition mod. 2
                   $\circ$ - concatenation
A conceptual block diagram of this ABP-model is shown in Fig. 3.
We consider both the SENDER as well as the RECEIVER to consist of a Processor and an Interface. This conceptual decomposition will be shown to be convenient for the modelling of the system behavior in case of faults. First, let us assume that the RECEIVER-Interface is capable of detecting any fault in the transmission of data messages (say by suitable time-out devices and error-detecting techniques). In case a data message is properly received, the RECEIVER-Interface will activate the transition ACCM (ACCEpt Message). Whereas, whenever a faulty situation is detected (excessive delay, error), the RECEIVER-Interface will activate the transition REJM (REJect Message) instead of ACCM. However, we need not model the details of the various types, nor the mechanisms of detecting them. The only relevant feature we include in our abstract model is the capability of the RECEIVER-Interface to perform non-deterministically: i.e., whenever place \( R_3 \) (see Fig. 2) is marked, either the transition ACCM or the transition REJM may be activated. At our level of abstraction, this may be considered as "arbitrary decision" of the RECEIVER-Interface.

Of course, in any actual implementation, this non-deterministic behavior will be replaced by a deterministic behavior, controlled by suitable fault-detection mechanisms. Evidently, any verification of the abstract model also applies to every possible implementation.

Similar arguments apply to the non-deterministic behavior of the SENDER-Interface (whenever place \( S_3 \) of Fig. 2 is marked).
Note that at this level of abstraction it is true that both the SENDER and RECEIVER can run arbitrarily without the other (e.g. the RECEIVER may make many loops for every message received, where only the first one will update CLIST). However, it is clear that when appropriate timeout parameters are assigned by a specific implementation, the behavior of the system becomes more coordinated. Verifying the protocol abstractly ensures correct behavior for any arbitrary timeout parameters.

The overall task to be performed by the ABP-System may be formulated as follows. The SENDER has available an infinite list of data messages (PLIST ∈ Σ^ω). It transmits these messages one by one:

\[ \text{PLIST}[1], \text{PLIST}[2], \ldots, \text{PLIST}[P], \ldots \]

where \( P \) denotes a pointer referring to the data message being currently transmitted. The RECEIVER accumulates the messages received into a finite list CLIST ∈ Σ^*.

Following the simple system of [BA-SC-WI], each transmission buffer is limited to one message, but there is no overwriting of consecutive messages, since (as is verified later) the SENDER produces and transmits a new message only after the previous message has been received and consumed by the RECEIVER.
4. MAIN RESULTS

In this section we formulate the main results of this paper, indicating both partial and full correctness of our ABP-System.

4.1 Partial Correctness

Let $\Sigma$ be an arbitrary finite alphabet, and let $\alpha$ denote an infinite string over $\Sigma$, i.e. $\alpha \in \Sigma^\omega$. $x \in \Sigma^*$ is a prefix of $\alpha$ iff $(\exists \beta \in \Sigma^\omega)(x\beta = \alpha)$. We denote by $\text{pref}(\alpha)$ the set of all prefixes of $\alpha$.

Let $q$ be an arbitrary state reachable from the initial state $q_0 = (M_0, d_0)$; i.e. $q_0 \cdot w \cdot q$ for some $w \in T^*$. We denote by $\text{LIST}(q)$ the value of the variable $\text{LIST}$ in state $q$. The following result summarizes the partial correctness of the ABP-System.

**Theorem 4.1.** (Partial Correctness) Let $q$ be a state of the ABP-System reachable from $q_0$. Then $\text{LIST}(q) \in \text{pref}(\text{PLIST})$.

Since $\text{LIST}$ specifies the list of messages consumed so far by the RECEIVER, the theorem ascertains that no loss, no error, no duplications and no misordering has happened during transmission.

4.2 Full Correctness

We have still to show that any number of data messages sent out by the SENDER will eventually be used by the RECEIVER. Of course, this feature cannot be expected to hold, if the system breaks down completely. At the level of abstraction of our model, this would mean that both Interfaces adhere to the "Principle of Fairness", i.e. they do not "permanently disregard an enabled transition".

We now proceed to define this concept formally. Let $E$ be an EPN, $q$ a state of $E$, and $\alpha$ an infinite sequence of transitions,
such that \( q[\alpha] \). We say that \( \alpha \) is fair with respect to \( q \) iff for every transition \( t \) the following statement is true: the number of occurrences of \( t \) in \( \alpha \) is finite, if and only if the set \( \{ x \in \text{pref}(\alpha) \mid q[xt] \} \) is finite. I.e. a transition \( t \) which is enabled an infinite number of times during \( \alpha \) must also be fired an infinite number of times.

The following result summarizes the full correctness of the ABP-System by ascertaining that CLIST is growing continuously.

**Theorem 4.2 (Full Correctness).** Let \( E_{\text{ABP}} \) be the ABP-System defined in Def. 3.2. Let \( \alpha \) be an infinite sequence of transitions, fair with respect to the initial state \( q_0 = (M_0,d_0) \). Let \( n \) be an arbitrary positive integer. Then there exists a state \( q \) of \( E_{\text{ABP}} \) such that \( q_0[\alpha] \not< q \) for some prefix \( x \) of \( \alpha \) and \( \text{CLIST}(q) \) has length \( n \).

Detailed proofs of the above results are given in the next section.

5. PROOF OF MAIN RESULTS

This section contains complete proofs of Theorems 4.1 and 4.2.

5.1 The Unified Transition Graph of the ABP-System

The data set \( D \) (see Def. 3.1) of the ABP-System, is actually the cartesian product of the domains (cf. Table 1) of the variables of the \( E_{\text{ABP}} \) (except \( \text{PLIST} \), which is constant).

\[
D = \{0,1\}^6 \times \Sigma^2 \times \Sigma^* \times \omega
\]

a typical element \( d \in D \) is denoted by:

\[
d = (SB, BL, RB, SA, AL, RA, ML, RM, CLIST, PT).
\]

Let \( V = \{0,1\}^6 \) be the projection of \( D \), formed by its first six components.
For any \( d \in D \), let \( v_d \) denote the (binary) vector of the first six elements of \( d \).

**Definition 5.1:** Let \( v \in V \) be a binary 6-tuple; then

\[
[v] \triangleq \{(M,d) \in \omega^P \times D | v_d = v\}.
\]

In fact, the vector \( v \) is composed of those data variables which take part in the condition function \( C \) (def. 3.1), and thus influence transition firing. Without considering the reachability of states we like to explore how the \( v \)-vectors may be changed during transition firing. For this purpose we define the "Unified Transition Graph", in which every vertex is a value of \( v \) and an edge exists from \( v \) to \( v' \), labeled by a transition \( t \), iff \( v \neq v' \) and if \( t \) is enabled in a state where \( v \) holds, a new state where \( v' \) holds is obtained by firing \( t \).

**Definition 5.2:** The Unified Transition Graph of \( E_{ABP} \) is a finite directed graph \( U = (V,E) \), where the set of vertices is \( V = \{0,1\}^6 \), and the set of edges is:

\[
E \triangleq \{(v,v') \in V \times V | v \neq v' \land (\exists q \in [v])(\exists q' \in [v'])(\exists t \in T)q[t> q']\}.
\]

Each edge is labelled by its corresponding transition \( t \).

Note, that since \( q, q' \) are not necessarily reachable, the graph \( U \) is not a homomorphism of the system's state-diagram, and some of the transitions in \( U \) are not actually firable (e.g. \( REJM \) in \([000111]\)), but this fact does not interfere with the verification, as may be observed further. However, \( U \) gives perfect information of the 'productive' behavior of the system in a significantly reduced form.
Following the above definition, U is easily formed (Fig. 4). The double arrow points to \( v_{d_0} \), and all the vertices with no incoming edges were omitted.

![Diagram](image)

**Figure 4** - The Unified Transition Graph of \( E_{ABP} \)

### 5.2 Partial Correctness

**Lemma 5.1:** In any firing sequence, starting from \((M_0, d_0)\), PROD and CONS appear alternatingly.

**Proof:** We show that at least one instance of CONS is contained between any two instances of PROD, and vice versa.

Let \( a \in T^* \) be a firing sequence, starting from \((M_0, d_0)\), and let \((M, d)\) be a state obtained by firing PROD once. Since for every \( d \), the firing of PROD causes a change in \( v_d \) (SB + SB 01), obviously there is no firing of PROD which has no respective edge, labelled PROD, in U. It is easily seen that any path in U from one PROD edge through another PROD edge must cross a CONS edge. Analogously, between two instances of CONS there is at least one of PROD. \(\square\)
Lemma 5.1 points out two important properties of the system behavior:

(1) there is no overwriting of messages, since a new message is never produced (an instance of PROD) before the previous message was consumed (an instance of CONS).

(2) no message is consumed more than once, since between the last consumption and the next one a production of a new message must occur.

Let \( \#(t,a) \) denote the number of instances of a transition \( t \in T \) in a firing sequence \( \alpha \in T^* \cup T^0 \).

**Lemma 5.2:** Let \( \alpha \in T^* \) be a firing sequence, such that \((M_0,d_0)[\alpha]=(M,d)\); then in state \((M,d)\):

(I) \( PT = \#(\text{PROD},a) \)

(II) \(|\text{CLIST}| = \#(\text{CONS},a)\)

**Proof:** Since at \((M_0,d_0)\):

(1) \( PT = 0 \)

(2) \( \text{CLIST} = \lambda \leftrightarrow |\text{CLIST}| = 0 \)

and since PROD is the only transition which changes PT (by incrementing it by 1) and CONS is the only transition which changes CLIST (particularly, by incrementing its length by 1), the lemma follows by induction. \( \square \)
We now proceed to prove Theorem 4.1, i.e., we show that

\[ \text{CLIST}(q) \in \text{pref}(\text{PLIST}) \]  

(Formula 5.1)

where \( q = (M, d) \) is a state reachable from \( q_0 = (M_0, d_0) \).

Remark: Let \( q, q' \) be arbitrary states, and \( \gamma \in T^* \) any firing sequence such that \( q[\gamma > q' \), then

\[ \#(\text{CONS}, \gamma) = 0 \Rightarrow \text{CLIST}(q) = \text{CLIST}(q'). \]

This is an immediate consequence of the definition of \( E_{ABP} \), since CONS is the only transition which changes CLIST.

Proof of Theorem 4.1: Let \( \alpha \in T^* \) be any firing sequence leading from \( q_0 = (M_0, d_0) \) to an arbitrary state \( q = (M, d): q_0[\alpha > q. \)

From Lemma 5.1 - \( \alpha = \alpha_0 \circ \text{PROD} \circ \alpha_1 \circ \text{CONS} \circ \cdots \circ \alpha_n \circ \text{PROD} \circ \alpha_n \circ \text{CONS} \circ \cdots \)

or particularly:

\[ q_0[\alpha_0 \circ \text{PROD} > q_0[\alpha_0 \circ \text{CONS} > q_1 \circ \alpha_1 \circ \text{CONS} > q_2 \circ \alpha_2 \circ \text{CONS} > q_3 \circ \alpha_3 \circ \text{CONS} > q_4 \circ \alpha_4 \circ \text{CONS} > q_5 \circ \alpha_5 \circ \text{CONS} > q_6 \circ \alpha_6 \circ \text{CONS} > \cdots q_n \]

where \( (\forall i \in \omega) \alpha_i, \alpha_i \in (T - \{\text{PROD}, \text{CONS}\})^* \).

Thus, it is sufficient to show that formula 5.1 holds for every \( q_n \).

We proceed by using induction on \( n \).

Basis From Table 1 - CLIST(q_0) = \( \lambda \in \text{Pref}(\text{PLIST}) \).

Induction Step: Denote \( \text{PLIST} = \sigma_1 \sigma_2 \sigma_3 \cdots \); then by the induction assumption, together with Lemma 5.2, CLIST(q_n) = \( \sigma_1 \sigma_2 \sigma_3 \cdots \sigma_n \) and PT(q_n) = \( n \). Again, by Lemma 5.2, PT(q_n) = \( n+1 \). PT remains at this value till state \( q_{n+1} \) is reached. From U,

\[ \hat{\alpha}_n = \beta_1 \circ \text{SEND} \circ \beta_2 \circ \text{ACC} \circ \beta_3, \]

where for \( j = 1, 2, 3 \) \( \beta_j \in (T - \{\text{PROD}, \text{CONS}\})^* \).
The following conclusions with respect to \( q_n \) are immediate consequences of the definition of \( E_{ABP} \):

1. Once SEND has been fired, \( ML \rightarrow P\text{LIST}[n+1] = \sigma_{n+1} \), and \( ML \) does not change, until state \( q_{n+1} \) is reached.

2. Then, once \( ACM \) has been fired, \( RM + ML = \sigma_{n+1} \) and remains at this value until \( q_{n+1} \) is reached.

3. Then, when CONS is fired, (in view of the remark above)

\[
\text{CLIST}(q_{n+1}) \leftrightarrow \text{CLIST}(q_n)^{\sigma_{RM}} = \sigma_1 \sigma_2 \ldots \sigma_n \sigma_{n+1} = \sigma_1 \sigma_2 \ldots \sigma_{n+1} \in \text{pref}(\text{PLIST}).
\]

\[\square\]

5.3 Fairness and Full Correctness

In Section 4.2 we have mentioned the reasons for including a "principle of fairness". This principle of fairness may be formulated as follows.

Let \( Q \) be the set of all reachable states of \( E_{ABP} \):

\[
Q \triangleq \{(M,d) \mid (\exists a \in T^*)(M_0.d_0)[a>(M,d)]\}.
\]

Definition 5.3:

\[
\text{fair} : T^\omega \times Q \rightarrow \{\text{true}, \text{false}\}
\]

is a predicate over all infinite firing sequences \( a \in T^\omega \) starting from any state \( q \in Q \), defined by:

\[
\text{fair}(a,q) \triangleq \forall t \in T \left[ \#(t,a) \text{ is finite} \Rightarrow \{x \in \text{pref}(a) \mid q[xt>] \text{ is finite} \} \right]
\]

i.e., any transition \( t \) has been fired a finite number of times during \( a \) iff it has been enabled only a finite number of times.*

Let \( \text{suff}(a) \) denote the set of all suffixes of a firing sequence \( a \), i.e.

\[
\text{suff}(a) \triangleq \beta \in T^* \mid \exists a \in T \cup T^\omega \left( \exists x \in \text{pref}(a) \right) \beta x = a.
\]

*) \( \triangleright \) in consequence, if it is infinitely often enabled it is infinitely often fired.
The following lemma ensures that every suffix of a fair sequence is fair too. This is an immediate consequence of Definition 5.3.

Lemma 5.3: Let \( a \in T^\omega \), \( \hat{a} \in \text{pref}(a) \), \( \beta \in \text{suff}(a) \), such that \( \hat{a}\beta = a \), \( q, q' \in Q \); then

\[
\text{fair}(a, q) \land q[\hat{a}\beta] = \text{fair}(\beta, q').
\]

"Partial deadlock" is a state in which a part of the system is active, while another part is deadlock. The following lemma verifies that no such situation could occur during a fair sequence, and both the SENDER and the RECEIVER are continuously active.

Lemma 5.4: Let \( T_S \triangleq \{\text{PROD, REPM, SEND, ACCA, REJA}\} \),

\[
T_R \triangleq T - T_S,
\]

and \( a \in T^\omega \) such that \( q_0[\alpha] \); then

1. \( (\exists \beta \in \text{suff}(a)) \beta \in T^\omega_S \Rightarrow \sim \text{fair}(a, q_0) \)
2. \( (\exists \beta \in \text{suff}(a)) \beta \in T^\omega_R \Rightarrow \sim \text{fair}(a, q_0) \).

Proof:

1. We have \((\forall x \in \text{pref}(a))(\exists t \in T_R)q_0[xt] > 0\), since (see Fig. 2)
   1. if \( M(R_1) = 1 \) then either CONS or REPA is enabled;
   2. if \( M(R_2) = 1 \) then SEND is enabled;
   3. if \( M(R_3) = 1 \) then both ACCM, REJM are enabled,

and obviously, for every \((M, d) \in Q:\ M(R_1) + M(R_2) + M(R_3) = 1\).

Thus, \((\forall t \in T_R)\) (the set \( \{x \in \text{pref}(a) | q_0[xt] \} \) is infinite). But

\[
(\exists \beta \in \text{suff}(a)) \beta \in T^\omega_S \Rightarrow
\]

\[
(\exists t \in T_R, \beta) \#(t, \beta) = 0 \Rightarrow
\]

\[
(\exists t \in T_R, \alpha) \sim \text{finite}.
\]
Now, assume \( \text{fair}(\alpha, q_0) \); it follows that \( \{ x \in \text{pref}(\alpha) | q_0[x^>] \} \) is finite, leading to a contradiction. Thus, \( \neg \text{fair}(\alpha, q_0) \).

(2) is analogous to (1).

The liveness of \( E_{ABP} \) is an immediate consequence of the proof of the last lemma. However, this does not require that the system is also "productive". The following lemma ensures that the system cannot proceed infinitely (during a fair sequence) without changing the value of \( v \), and therefore, must proceed along a path in the graph \( U \).

Lemma 5.5: Let \( q_0, q_1, q_2, \ldots \) be a consecutive sequence of reachable states:

\[
q_0[t_0^>] q_1[t_1^>] \ldots
\]

and assume \( \text{fair}(\alpha, q_0) \), where \( \alpha = t_0 t_1 t_2 \ldots \).

Then

\[
(\forall q_i = (M_i, d_i))(\exists q_j = (M_j, d_j))(j > i \land v_{d_j} \neq v_{d_i})
\]

Proof: Assume that \( (\exists q_i = (M_i, d_i))(\forall q_j = (M_j, d_j), j > i) v_{d_j} = v_{d_i} = v \).

We denote: \( q_0[\hat{a} q_i[\hat{b}^>] \) where \( \hat{a} = a, \hat{a} = t_0 t_1 \ldots t_{i-1} \).

Since each firing of PROD or CONS always changes \( v_d \), \( \#(\text{CONS}, \beta) = \#(\text{PROD}, \beta) = 0 \).

By Lemma 5.3, \( \text{fair}(\beta, q_1) = \text{true} \), and by Definition 5.3 neither CONS nor PROD is enabled during \( \beta \). Thus, (see Fig. 2), the following must hold for \( v \): \( \text{RB} = \text{SA} \land \text{RA} \neq \text{SB} \).

By Lemma 5.4; \( \#(\text{SEND}, \beta) > 0 \) and \( \#(\text{ACK}, \beta) > 0 \), and since \( v \) is not changed during \( \beta \), \( \text{BL} = \text{SB} \land \text{AL} = \text{SA} \). Summarizing the above, \( v \) must have the form \( [x y y y x] \), where \( x, y \in \{0, 1\} \).
There are only 4 possibilities for $v$: (1) [111110]
(2) [000001]
(3) [110000]
(4) [001111] .

We prove Case (1); the others are analogous.

We have (see Fig. 4), $\#(ACCA, B) = 0$ (otherwise $v$ would be changed).

By Lemma 5.4 and Definition 5.3, $\#(REJA, B) = \infty$, but since $REJA/ACCA$
are always enabled together, it follows that $ACCA$ has been enabled an
infinite number of times during $\beta$, and has never been fired, in
contradiction to $fair(\beta, q_i)$. Thus, our initial assumption cannot hold.

We now proceed to prove Theorem 4.2, which may be reformulated
as follows:

$(\forall n \in \omega)(\forall a \in T^\omega)$

$fair(a, q_0) \Rightarrow (\exists \hat{a} \in pref(a))(\exists q \in Q)q_0[\hat{a} > q \land |CLIST(q)| = n]$ .

**Proof:** By Lemma 5.2 $|CLIST(q)| = \#(CONS, \hat{a})$; thus it is sufficient
to prove that for every infinite firing sequence $a$, $\#(CONS, a) = \infty$.

By Lemma 5.5, $a$ contains an infinite path in $V$, and such a path
contains CONS infinitely many times, unless it circles infinitely in
one of the "shorter" loops, namely:

1. [11111] $\xrightarrow{REJA} \xleftarrow{ACCA}$ [111110]
2. [001111] $\xrightarrow{ACCM} \xleftarrow{REJM}$ [000111]
3. [000001] $\xrightarrow{ACCA} \xleftarrow{REJA}$ [000000]
4. [111000] $\xrightarrow{REJM} \xleftarrow{ACCM}$ [110000]
But this cannot be the case.

Again, we prove this only for loop (1), the proofs for the other loops are analogous. Suppose that the path enters the loop. Once ACCA has been fired, \([111111]\) is reached, while \(M(S_1) = 1\) (see Fig. 2). In this situation PROD is enabled, while REPM is not, and this cannot be changed until PROD is fired, which would eventually occur, due to Lemma 5.4. But, when PROD is fired, the path leaves the loop.

6. RELATED WORKS

Various other models of protocols, distinguish, similar to our model, between control and data.

The control is represented by means of \(\mu\)CLA-graphs in [PO-FA], [RA-ES], and by modified Petri-Nets in [SYM], [BO-GE]. In addition, our model clearly distinguishes between the \(local\) control represented by a simple Petri-Net, and the \(inter-party\) control, by means of the corresponding extension of the above-mentioned simple Petri Net.

The EPN, used in our model, is quite close to the NPN of [SYM], and our proof techniques could also be applied to NPN's.

Our \(E_{ABP}\) contains all the features included in the complex graph model of [PO-FA], except "bounded-looping" which we replace by "finite-looping" in view of our "principle of fairness". Moreover, their model contains the assumption that every party may "sense" the line to check whether a message (or ack.) has been received, while our model is more abstract. Neither does our model assume any particular timeout mechanism, as explained in Section 3.2. The actual timeout parameters
may be assigned by specific implementation.

The approaches of [BR-DR], [HA-OW], [SUN 80] are significantly different from ours, and involve more complex formal verification techniques.

ACKNOWLEDGEMENT

The authors are grateful to Dr. C. Sunshine for his valuable comments which helped to improve the paper.
REFERENCES


REFERENCES (cont'd)
