Petri Net Based Verification of Distributed Algorithms: An Example*

Ekkart Kindler, Wolfgang Reisig, Hagen Völzer, Rolf Walter
Humboldt-Universität zu Berlin†
Institut für Informatik, D-10099 Berlin,Germany

May 22, 1996

Abstract

A technique to describe and to verify distributed algorithms is suggested. This technique (based on Petri nets) reduces the modelling- and analysis effort to a reasonable expenditure. The paper outlines the technique along a typical network alogrithm, the *echo algorithm*.

Keywords: Modelling, Correctness, Petri Nets, Verification Techniques, Temporal Logic.

1 Introduction

A wide class of basic sequential algorithms (such as searching and sorting) can optimally be formulated in Pascal like programming notation and verified in the style of Hoare logic. A likewise generally accepted, integrated pair of techniques to formulate and verify *distributed* algorithms still remains to emerge. This paper is intended to contribute to this purpose.

A distributed algorithm is adequately represented iff the employed operational primitives focus the essentials of its algorithmic idea. Hence the adequacy of a representation technique can best be demonstrated by help of a typical example. We have chosen Petri nets as a modelling technique with atomic actions that receive, synchronize and forward messages. The representation technique is demonstrated by help of the well-known echo algorithm [3].

Likewise a distributed algorithm is adequately verified iff the proof argues tightly along the operational primitives and the structure of the algorithm's representation (e.g. [2, 8, 9, 13, 14]). We suggest a proof style which exploits standard techniques from Petri net theory: Invariant properties of the echo algorithm are proven by help of P–invariants of the corresponding Petri net.

The echo algorithm is a typical algorithm operating on an arbitrary, connected network of agents. A distinguished agent called the *initiator* eventually takes a decision based on the participation of all agents without having global knowledge about the network. To do so, the initiator has to inform every other agent and decides only when he is sure that every other agent has received and acknowledged this information.

*supported by the DFG–projects "Verteilte Algorithmen" and "Konsensalgorithmen"
†e–mail {kindler,reisig,voelzer,walter}@informatik.hu-berlin.de

The echo algorithm is used for quite a lot of purposes, e.g. as part of termination detection [4, 12] or for finding a minimal spanning tree [5]. It is also known as PIF-algorithm [11] and used for general synchronization tasks in networks. Its intuitive idea is simple, but formal proofs (e.g.[1]) are rarely found.

The paper is organized as follows. In the first section we introduce the algorithm as well as the modelling technique. The essential properties are stated in terms of classical linear time temporal logic. Section 2 introduces some notations. The essence of the used proof technique is presented in Section 3. Section 4 contains the verification of the echo algorithm. An appendix elaborates the formal basis of the modelling and verification technique.

2 The echo algorithm

In this section we provide a Petri net representation of the echo algorithm. This serves to recall the algorithm, as well as to informally introduce high-level Petri nets.

2.1 Modelling the echo algorithm

The echo algorithm works on a connected *network of agents*. The finite set of agents is denoted by A and we assume that there is one distinguished agent $i \in A$ called *initiator*. Two agents $x, y \in A$ may be connected by a bidirectional *channel*. If there is a channel from x to y we say x and y are *neighbours*. Figure 1 shows an example of a network of agents with $A = \{\mathsf{i}, \mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{d}, \mathsf{e}\}$. The connections between the agents are represented as a set of

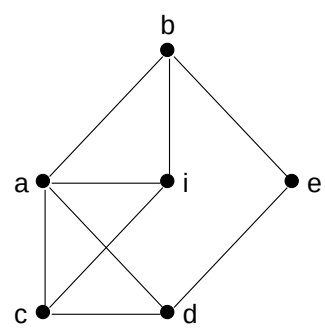

Figure 1: A network of agents

pairs $N \subseteq A \times A$, where a channel from x to y is represented as a pair (x, y). Note, that for $(x, y) \in N$ we also have $(y, x) \in N$, since we assume that channels are bidirectional. For example, we have $(\mathsf{a}, \mathsf{i}), (\mathsf{i}, \mathsf{a}) \in N$ for the network of Fig. 1, but $(\mathsf{i}, \mathsf{d}) \notin N$. To sum up, the static structure of the network of agents is represented as a set of agents A and a set of edges N. These two sets form the *domain* of the forthcoming high-level Petri net.

Next we describe the dynamic behaviour of the echo algorithm by a Petri net. A Petri net (for an example see Fig. 2) consists of a set of *places*, which are graphically represented by ellipses, and a set of *transitions*, which are represented by squares. Places and transitions are related to each other by *arcs*. A *state* of a Petri net is represented by *tokens* on the different places of the Petri net. A state is changed by an *occurrence* of a transition. In conventional Petri nets tokens are not distinguishable and considered to be black. Here, we use *high-level Petri nets* where tokens are elements of a particular *token domain*.

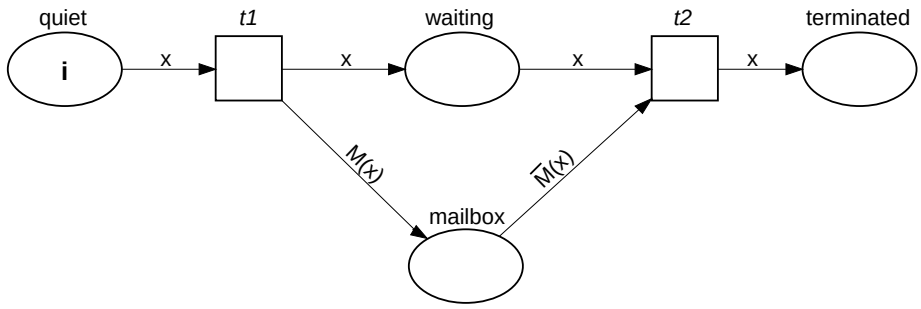

Figure 2: Behaviour of the initiator

For modelling the behaviour of an agent $x \in A$ we distinguish two cases: x is the initiator or x is one of the other agents. The Petri net model for x being the initiator is shown in Fig. 2. The initiator starts the algorithm by sending a message to each of his neighbours. Sending a message from agent x to agent y is modelled by putting token (y, x) on place mailbox — note, that we set y in the first place because the message is put to y's mailbox. Then, sending a message from x to each of his neighbours $y_1, \ldots, y_n$ is formalized by the arc-inscription $M(x) = [(y_1, x), \ldots, (y_n, x)]$. After sending a message to each of his neighbours the initiator waits (at place waiting) until he receives a message from each of his neighbours and, then, terminates. According to the above interpretation of tokens (x, y) on place mailbox, an agent x has received messages from each of his neighbours $y_1, \ldots, y_n$, when all tokens $\overline{M}(x) = [(x, y_1), \ldots, (x, y_n)]$ are present on mailbox. Thus, the arc-inscription $\overline{M}(x)$ formalizes that the initiator must wait for a message from each of his neighbours.

The Petri net modelling the behaviour of the other agents is shown in Fig. 3. When an agent x receives a message of some agent y — represented by a token (x, y) on place mailbox — he sends a message to all other neighbours. Then, we call y the *father* of x. Sending these messages is modelled by the arc-inscription $M(x) - (y, x)$. Sending the message to his father is delayed (cf. place pending-with) until he has received a message from all other agents (arc-inscription $\overline{M}(x) - (x, y)$). Upon reception of all these messages, x returns a message to his father y and terminates at place accepted.

Now, we have formalized the behaviour of all agents. To get the complete behaviour of the network of agents we just have to combine the two Petri nets from Fig. 2 and 3 to

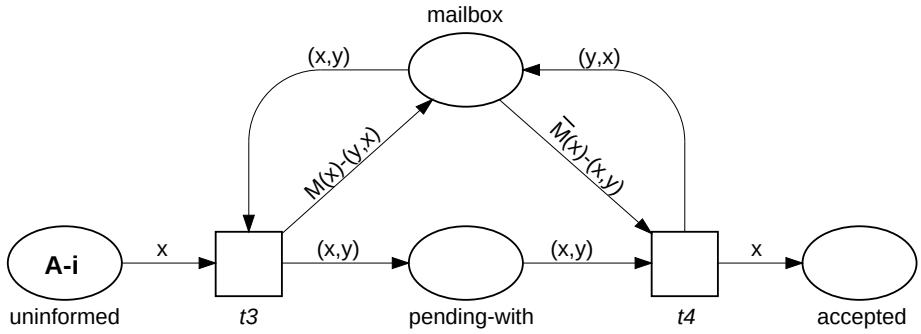

Figure 3: Behaviour of the other agents

a single Petri net as shown in Fig. 4. Here, we have used Petri nets informally, only. A

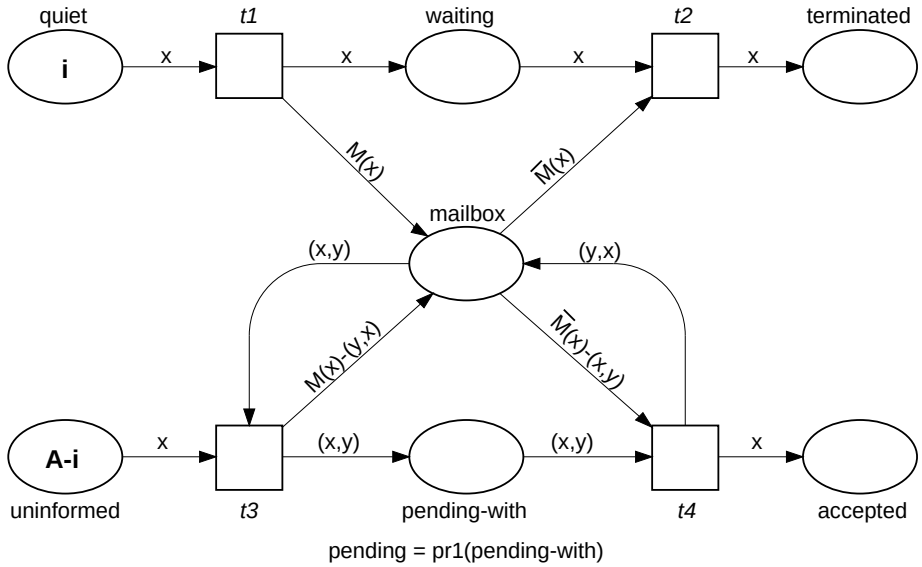

Figure 4: Petri net Σ: the echo algorithm

concise definition of high-level Petri nets is given in the appendix. The mathematical theory of high-level Petri nets can be found in [6].

2.2 Properties of the echo algorithm

Now, we will formalize the essential properties of the echo algorithm, which will be verified in Sect. 4. Again, we introduce our formalism in an informal way. As usual, we distinguish *safety* and *liveness properties*.

The first property states that the initiator will eventually terminate, which is a liveness property. More formally, this means that each *run* of the echo algorithm has a state with a token i at place terminated. This property is represented in a temporal logic-like notation by

$$\Diamond\ \mathsf{terminated}(i) \tag{L}$$

The second property states that the initiator will not terminate before all other agents have accepted, which is a safety property. We represent this property by

$$\forall x \in A \setminus \{i\} \quad \Box\ \mathsf{terminated}(i) \rightarrow \mathsf{accepted}(x) \tag{S}$$

3 Some verification techniques

The main purpose of this paper is to demonstrate the application of Petri nets for verifying distributed algorithms. Before we will verify the echo algorithm in Sect. 4, we informally introduce some specific Petri net verification techniques and some appropriate notations. Again, a formal presentation can be found in the appendix.

The current *state* of a high-level Petri net represents, which *tokens* are present at the different places of the net. Formally, a state is a mapping from the set of places to multisets over a given domain. Therefore, we introduce some notations for multisets. A finite multiset is represented as $[a_1, a_2, \ldots, a_n]$, where multiple occurrences of a single element are allowed (and relevant). In particular $[a]$ denotes the multiset with exactly one element a; $[]$ represents the empty multiset. For a multiset m and some element a, we denote the number of occurrences of a in m by $m[a]$. The addition of two multisets m_1 and m_2 is denoted by $m_1 + m_2$ and defined by $(m_1 + m_2)[a] = m_1[a] + m_2[a]$ for each element a.

Now, for a given state the name of a place in some expression or proposition can be interpreted as the multiset of tokens which currently reside at this place. For example, in the initial state the simple expression quiet denotes the multiset $[i]$. This way, we can argue on states of a Petri net. For example, $\mathsf{quiet} + \mathsf{waiting} + \mathsf{terminated} = [i]$ denotes all states in which there is exactly one token i on one of the places quiet, $\mathsf{waiting}$, or $\mathsf{terminated}$. From propositions on states we build two kinds of temporal propositions: for a proposition p the temporal proposition $\Box\ p$ means, that each reachable state of a Petri net satisfies the proposition p. When $\Box\ p$ is satisfied by a Petri net, we say, that p is an *invariant* of this Petri net. The temporal proposition $\Diamond\ p$ means, that in each run there exists a state which satisfies p. The symbols $\Box$ and $\Diamond$ are called 'always' and 'eventually' operator, respectively.

For example, the echo algorithm satisfies $\Box\ \mathsf{quiet} + \mathsf{waiting} + \mathsf{terminated} = [i]$. This can easily be verified by a standard technique in Petri net theory, called *P-invariant*-analysis. A P-invariant is an expression, which is built from some place names of the Petri net, such that an occurrence of a transition does not change the value of this expression. For example, $\mathsf{quiet} + \mathsf{waiting} + \mathsf{terminated}$ is a P-invariant. To verify a P-invariant we have to check the validity of one equation for each transition of the Petri net. For a transition t the left-hand side of the corresponding equation is the P-invariant expression, where each place name p is substituted by the arc-inscription of the arc from p to t (if there is no arc we substitute $[]$); the right-hand side of the equation is the P-invariant expression, where each place name is substituted by the arc-inscription of the arc from t to p. For the P-invariant $\mathsf{quiet}+\mathsf{waiting}+\mathsf{terminated}$ and transition t_1 and t_2 we get the equations $[x]+[]+[] = []+[x]+[]$ and $[] + [x] + [] = [] + [] + [x]$ respectively, which are obviously valid. For t_3 and t_4 we get the trivial equation $[] + [] + [] = [] + [] + []$.

From this P-invariant and the initial state, we immediately get $\Box\ \mathsf{quiet} + \mathsf{waiting} + \mathsf{terminated} = [i]$. Now, we can project this equality on multisets to a single element, which yields a proposition on numbers. For example, the projection for the initiator i reads:

$$\Box\ \mathsf{quiet}[i] + \mathsf{waiting}[i] + \mathsf{terminated}[i] = 1$$

This invariant will be used in the next section.

Although, many interesting invariants can be proved by P-invariants, some cannot be proven this way. In this case we use the classical technique of *assertional reasoning*: for proving an invariant p we check two conditions:

1. Proposition p must hold initially.
2. When p holds before the occurrence of a transition, then p holds after the occurrence of this transition, too.

Finally, we introduce an abbreviation that we have already used in the formalization of the properties of the echo algorithm. When we state that some multiset m contains at least the element a, we write $m(a)$ — this use of parentheses indicates that multiset m is interpreted as a predicate. Formally, $m(a)$ is just an abbreviation for $m[a] \geq 1$. Thus, $\mathsf{terminated}(i) \rightarrow \mathsf{accepted}(x)$ is an abbreviation for $\mathsf{terminated}[i] \geq 1 \rightarrow \mathsf{accepted}[x] \geq 1$.

4 Verification of the echo algorithm

This section will give a formal proof of the above properties (S) and (L). We start with some basic properties immediately derivable from the Petri net.

4.1 Basic properties of the echo algorithm

First we observe that tokens on each place always belong to a restricted subset of the token domain. For example, the place uninformed never contains a token i, which represents the initiator. So, we assign to each place of the net a subset of the token domain of the Petri net, which we call the *type* of the corresponding place. We call such an assignment a *type invariant* of the Petri net iff in each reachable state of the Petri net each place contains only tokens of its type. Table 1 shows a type invariant for Σ. This type invariant is easily verified by assertional reasoning. Next we state some more invariants of the algorithm which are

quiet	$\{i\}$	uninformed	$A \setminus \{i\}$
waiting	$\{i\}$	pending-with	$N \cap ((A \setminus \{i\}) \times A)$
terminated	$\{i\}$	accepted	$A \setminus \{i\}$
mailbox	N		

Table 1: A type invariant for Σ

easily derived from the model by classical Petri net P-invariants.

The first invariant states that the initiator is always in exactly one of the three local states quiet, waiting or terminated. We denote this by:

$$\Box\ \mathsf{quiet}[i] + \mathsf{waiting}[i] + \mathsf{terminated}[i] = 1 \tag{1}$$

An analogous statement holds for all other agents of the network Each of them is always in exactly one of the three local states uninformed, pending[1] or accepted:

$$\forall x \in A \setminus \{i\} \quad \Box\ \mathsf{uninformed}[x] + \mathsf{pending}[x] + \mathsf{accepted}[x] = 1 \tag{2}$$

Finally, we derive from another P-invariant a more sophisticated invariant property covering the life-cycle for messages. Each possible message $(x, y) \in N$ is in exactly one of the following five local states:

1. (x, y) is not yet sent, i.e. (x, y) is virtually still with the sender, i.e. either $\mathsf{quiet}(y)$ or $\mathsf{uninformed}(y)$ holds.
2. (x, y) is in the mailbox of x, i.e. $\mathsf{mailbox}(x, y)$ holds.
3. (x, y) is received as a wake-up-message, i.e. $\mathsf{pending\text{-}with}(x, y)$ holds.
4. (x, y) is put aside for later sending, i.e. $\mathsf{pending\text{-}with}(y, x)$ holds.

[1] Formally, the multiset pending is defined (as indicated in Fig. 4) as the projection on the first component applied to the elements of the multiset pending-with. Thus, pending is a multiset over A and will be interpreted as a predicate over agents.

5. (x, y) is received as a non-wake-up-message, i.e. either $\mathsf{terminated}(x)$ or $\mathsf{accepted}(x)$ holds.

This property of messages is represented by the proposition:

$$\forall (x,y) \in N \quad \Box\ \mathsf{quiet}[y] + \mathsf{uninformed}[y] + \mathsf{mailbox}[(x,y)] + \mathsf{pending\text{-}with}[(x,y)] + \mathsf{pending\text{-}with}[(y,x)] + \mathsf{terminated}[x] + \mathsf{accepted}[x] = 1 \tag{3}$$

This invariant covers almost the whole net. Note, that invariant (3) has already some interesting implications. For example, (3) shows that it never happens that one of two neighbours has already accepted while the other one is still uninformed.

4.2 The safety property of the echo algorithm

At first we show the validity of property (S). A key argument for the whole proof is derivable from looking at the place pending-with. This place always contains pairs of agents. In the sequel, this set of pairs will be considered as a graph, called the *pending-graph*. Due to Table 1 the pending-graph is a subgraph of N. We prove now that the pending-graph is always acyclic:

1 Lemma In Σ holds:

$$\Box\ \mathsf{acyclic(pending\text{-}with)} \tag{4}$$

Proof: by assertional reasoning

1. (4) holds initially, since pending-with is initially empty.
2. Now we assume a state where the pending-graph is acyclic. We have to show that this doesn't change when a transition occurs.
   (a) Occurrence of transitions t1 and t2 does not change the pending-graph.
   (b) Occurrence of t4 leaves the pending-graph acyclic, since t4 retracts edges only.
   (c) We consider now occurrence of t3 in mode $[x = a, y = b]$. This event is enabled only if $\mathsf{uninformed}(a)$ holds. Then, due to (3) and Table 1 proposition $\neg\exists z\ \mathsf{pending\text{-}with}(z, a)$ must hold as well. That means that there is no edge directed to a in the pending-graph when t3 occurs in mode $[x = a, y = b]$. Thus, occurrence of t3 cannot establish a cycle in the graph.

$\Box$

To show that the initiator terminates only if all other agents have accepted it suffices to show that if the initiator is terminated then no agent is still uninformed or pending. We prove the latter with the next two lemmata.

2 Lemma In Σ holds

$$\Box\ \mathsf{pending}(x) \rightarrow \neg\mathsf{terminated}(i) \tag{5}$$

Proof: First we gain from (3):

$$\forall (x,y) \in N \quad \Box\ \mathsf{pending\text{-}with}(x,y) \rightarrow (\neg\mathsf{uninformed}(y) \wedge \neg\mathsf{accepted}(y)) \tag{6}$$

and from (2):

$$\forall y \in A \setminus \{i\} \quad \Box\ \neg(\mathsf{uninformed}(y) \wedge \neg\mathsf{accepted}(y)) \rightarrow \mathsf{pending}(y) \tag{7}$$

We derive from (6) and (7):

$$\forall (x,y) \in N \quad \Box\ \mathsf{pending\text{-}with}(x,y) \rightarrow (\mathsf{pending}(y) \vee y = i) \tag{8}$$

Proposition (8) means the following: For each pending agent x another agent y, who is the father of x, is pending as well if y is not the initiator. By repetition of this argument we obtain a sequence of pending agents. That sequence is by construction a path of the pending-graph and is therefore acyclic by lemma 1. Moreover, the pending-graph and therewith every path in it is finite. Due to (8) an end of the sequence must be the initiator. Therefore, the following holds:

$$\Box\ \mathsf{pending}(x) \rightarrow \exists z\ \mathsf{pending\text{-}with}(z,i) \tag{9}$$

We know by (3) that $\mathsf{pending\text{-}with}(z,i)$ implies $\neg\mathsf{terminated}(i)$. Thus, we obtain (5) immediately from (9). $\Box$

3 Lemma In Σ holds

$$\Box\ \mathsf{uninformed}(x) \rightarrow \neg\mathsf{terminated}(i) \tag{10}$$

Proof: by induction over reachability of nodes from i in (A,N), i.e.

1. (10) holds for $x = i$ due to Table 1: The initiator is never uninformed.
2. We assume now $(a,b) \in N$ and the validity of (10) for $x = a$. Assuming $\mathsf{uninformed}(b)$ we conclude $\neg\mathsf{accepted}(a)$ by (3). By (2) we distinguish three cases:
   (a) $a = i$: Then, $\neg\mathsf{terminated}(i)$ follows by (3) (taking $x = i$ and $y = b$).
   (b) $\mathsf{pending}(a)$: Then, $\neg\mathsf{terminated}(i)$ follows by lemma 2.
   (c) $\mathsf{uninformed}(a)$: Then, $\neg\mathsf{terminated}(i)$ follows by induction hypothesis.

Since the graph (A,N) is connected, the induction reaches every agent in the network. $\Box$

4 Theorem Petri net Σ satisfies property (S).

Proof: Let $x \in A \setminus \{i\}$. We fix a reachable state where $\mathsf{terminated}(i)$ holds. By lemmata 2 and 3 we get $\neg\mathsf{uninformed}(x) \wedge \neg\mathsf{pending}(x)$. Then, due to (1) $\mathsf{accepted}(x)$ must hold. $\Box$

4.3 The liveness property of the echo algorithm

Proposition (L) will be proven in two steps. First we show that the algorithm terminates. We define a state proposition $\mathsf{lockout}_\Sigma$ that holds in every state which doesn't enable any transition of the Petri net Σ. Then, termination is denoted by

$$\Diamond\ \mathsf{lockout}_\Sigma \tag{11}$$

The proposition

$$\Box\ \mathsf{lockout}_\Sigma \rightarrow \mathsf{terminated}(i) \tag{12}$$

expresses that whenever the Petri net terminates, the initiator must be terminated. This will be shown as a second step.

We show $\Diamond\ \mathsf{lockout}_\Sigma$ by help of a *descending variant*. In analogy to P-invariants, a descending variant is a state expression which contains names of places of the net. Its value is a natural number. Each occurrence of a transition of the Petri net strictly decreases that value. Thus, transition occurrences can happen only finitely many times. A suitable descending variant for Σ is the simple expression

$$2||\mathsf{quiet}|| + 1||\mathsf{waiting}|| + 2||\mathsf{uninformed}|| + 1||\mathsf{pending\text{-}with}||$$

where $||m|| \in \mathbb{N}$ denotes the cardinality of the multiset m. This descending variant has an intuitive interpretation: Tokens which represent agents are restricted to flow from left-hand side to right-hand side of the net, where finally no more flow is possible. Formal verification of that descending variant is similar to the verification of P-invariants as described in section 2.2. A concise presentation is given in the appendix.

In order to prove (L) proposition (12) remains to be proven. We start with two lemmata which state that in case of termination the places uninformed and pending-with are empty.

5 Lemma In Σ holds:

$$\Box\ \mathsf{uninformed}(x) \rightarrow \neg\mathsf{lockout}_\Sigma \tag{13}$$

Proof: again by induction over reachability in (A, N):

1. (13) holds for $x = i$, because the initiator is never uninformed due to Table 1 .
2. We assume now $(a, b) \in N$ and the validity of (13) for $x = a$. Assuming $\mathsf{uninformed}(b)$ we conclude $\neg\mathsf{terminated}(b)$ by Table 1, $\neg\mathsf{accepted}(b)$ by (2) and Table 1, furthermore $\neg\mathsf{pending\text{-}with}(a, b)$ and $\neg\mathsf{pending\text{-}with}(b, a)$ by (3). Again by (3) we obtain $\mathsf{quiet}(a) \vee \mathsf{mailbox}(b, a) \vee \mathsf{uninformed}(a)$.
   (a) In case of $\mathsf{quiet}(a)$, transition t1 may occur in mode $[x = a]$. (Remember that enabling of any transition in any mode implies $\neg\mathsf{lockout}_\Sigma$.
   (b) In case of $\mathsf{mailbox}(b, a)$, transition t3 may occur in mode $[x = b, y = a]$, since we assumed $\mathsf{uninformed}(b)$.
   (c) In case of $\mathsf{uninformed}(a)$, we conclude $\neg\mathsf{lockout}_\Sigma$ by induction hypothesis.

Since the graph (A, N) is connected, the induction reaches every agent in the network. $\Box$

6 Lemma In Σ holds:

$$\Box\ \mathsf{pending}(x) \rightarrow \neg\mathsf{lockout}_\Sigma \tag{14}$$

Proof: Assume a reachable state where at least one agent is pending, i.e. the pending-graph is not empty. Due to lemma 1 we can find always a leaf in that graph, i.e. an agent b with the property $\exists c : \mathsf{pending\text{-}with}(b,c) \wedge \forall x : \neg\mathsf{pending\text{-}with}(x,b)$.

From $\mathsf{pending\text{-}with}(b,c)$ we conclude $\neg\mathsf{terminated}(b), \neg\mathsf{accepted}(b)$ by (3) and for $x \neq c$ we conclude $\neg\mathsf{pending\text{-}with}(b,x)$ by (2). Then, by (3) we gain $\mathsf{quiet}(x) \vee \mathsf{mailbox}(b,x) \vee \mathsf{uninformed}(x)$ for those x with $(x,b) \in N$ and $x \neq c$. Again we distinguish:

1. In case of $\mathsf{quiet}(x)$ for *any* x, t1 may occur.
2. In case of $\mathsf{uninformed}(x)$ for *any* x, we conclude $\neg\mathsf{lockout}_\Sigma$ by lemma 5.
3. In case of $\mathsf{mailbox}(b,x)$ for *all* x, t4 may occur in mode $[x = b, y = c]$, since we assumed pending-with(b,c).

Therewith (14) is proven. $\Box$

7 Theorem Petri net Σ satisfies property (L).

Proof: Obviously, (11) and (12) imply (L), where (11) is already shown. Thus, we prove (12) now. We assume a state where $\mathsf{lockout}_\Sigma$ holds. In such a state the place quiet is obviously empty and due to lemma 5 and lemma 6 the places pending-with and uninformed are empty as well. To sum it up we state:

$$\Box\ \mathsf{lockout}_\Sigma \rightarrow \neg\mathsf{quiet}(x) \wedge \neg\mathsf{uninformed}(x) \wedge \neg\mathsf{pending}(x) \tag{15}$$

Especially $\neg\mathsf{quiet}(i)$ holds when $\mathsf{lockout}_\Sigma$ holds and by (1) we gain:

$$\Box\ \mathsf{lockout}_\Sigma \rightarrow \mathsf{waiting}(i) \vee \mathsf{terminated}(i) \tag{16}$$

Furthermore we know from Table 1 that the initiator is never in the local state accepted. Thus, we gain by (3) from (15) for arbitrary neighbours x of the initiator:

$$\forall (i,x) \in N \quad \Box\ \mathsf{lockout}_\Sigma \rightarrow \mathsf{mailbox}(i,x) \vee \mathsf{terminated}(i) \tag{17}$$

Combining (16) and (17) we gain:

$$\forall (i,x) \in N \quad \Box\ \mathsf{lockout}_\Sigma \rightarrow (\mathsf{mailbox}(i,x) \wedge \mathsf{waiting}(i)) \vee \mathsf{terminated}(i) \tag{18}$$

In case of $\mathsf{waiting}(i) \wedge \mathsf{mailbox}(i,x)$ for all neighbours x of the initiator, t2 may occur which contradicts $\mathsf{lockout}_\Sigma$. Therefore, at least for one x this is not the case and therewith $\mathsf{terminated}(i)$ follows by (18). $\Box$

With theorem 7 we have finished the proof of the liveness property (L) and the verification of the echo algorithm is complete.

5 Conclusion

The echo-algorithm was discussed as an example showing that Petri nets are a feasible model for the description of distributed algorithms. The correctness proof was carried out by help of formal verification techniques that are tightly connected to the employed operational model.

The presented proof follows the general aim of the approach to diminish the frequently observed trade-off between technical precision and intuitive simplicity. The use of Petri nets supports this aim, abstracting from a concrete programming notation together with the use of techniques based on multisets and state expressions over the places of the Petri net.

Studies have shown that a lot of distributed algorithms fit to the proposed proof style, e.g. [7, 10].

References

[1] L. Bougé. Modularité et Symétrie pour les Systèmes Répartis; Application au Langage CSP. Rept. No. 87-2, LIENS, Univ. Paris 7, 1987.

[2] K. M. Chandy and J. Misra. *Parallel Program Design: A Foundation.* Addison-Wesley, 1988.

[3] E.J.H. Chang. Echo algorithms: Depth parallel operations on general graphs. *IEEE Transactions on Software Engineering*, SE-8(4):391–401, 1982.

[4] E. W. Dijkstra and C. S. Scholten. Diffusing computations. *Information Processing Letters*, 1981.

[5] R. G. A. Gallager, P. Humblet, and M. Spira. A distributed algorithm for minimum-weight spanning trees. *ACM Transactions on Programming Languages and Systems*, 5(1):66–77, 1983.

[6] K. Jensen. *Coloured Petri Nets*, Volume 1 of *EATCS Monographs on Theoretical Computer Science.* Springer-Verlag, 1992.

[7] E. Kindler and R. Walter: "'Message Passing Mutex"'. In Desel, J. (Ed.): *Proceedings of STRICT.* Workshops in Computing. Springer-Verlag London: 1995: 205–219.

[8] L. Lamport. A temporal logic of actions. SRC Research Report 57, Digital Equipment Corporation, Systems Research Center, April 1990.

[9] N. Lynch and M. Tuttle. An introduction to input/output automata. *CWI-Quarterly*, 3(2):219–246, 1989.

[10] W. Reisig: "'Petri Net Models of Distributed Algorithms"'. In Leeuven, Jan van (Ed.): *Computer Science Today. Recent Trends and Developments.* Lecture Notes in Computer Science **1000**. Berlin: Springer-Verlag: 1995: 441–454.

[11] A. Segall. Distributed network protocols. *IEEE Transactions on Information Theory*, IT29-1:23–35, 1983.

[12] N. Shavit and N. Francez. A new approach to detection of locally indicative stability. In L. Kott, editor, *Proc. of th 13th ICALP*, volume 226 of *LNCS*, pages 344–358. Springer-Verlag, 1986.

[13] F. Stomp and W. P. de Roever. A principle for sequential phased reasoning about distributed algorithms. *Formal Aspects of Computing*, 6:716–737, 1994.

[14] G. Tel. *Topics in Distributed Algorithms*, volume 1 of *Cambridge International Series on Parallel Computation*. Cambridge University Press, 1991.

Appendix

A: System model

There are many different ways to formalize high-level Petri nets. Here we choose a notation which allows for syntactic manipulation of *terms* to derive properties from a given algebra. We start with some notations for multisets, algebras, and terms. High-level Petri nets and their dynamic behaviour follow.

Multisets For a countable set A, a mapping $m : A \to \mathbb{N}$ is called a *multiset over* A. The set of all multisets over A is denoted $\mathbf{MS}(A)$. For better readability we write $m[a]$ instead of $m(a)$ for a multiset $m \in \mathbf{MS}(A)$ and $a \in A$. The *empty multiset* $\mathcal{O}$ is defined by $\mathcal{O}[a] = 0$ for each $a \in A$. The *cardinality* $\|m\|$ of a multiset $m \in \mathbf{MS}(A)$ is $\|m\| \mathrel{\widehat{=}} \sum_{a \in A} m[a]$. A multiset $m \in \mathbf{MS}(A)$ is *finite* iff $\|m\| \in \mathbb{N}$. The set of *finite multisets* over A is denoted by $\mathbf{MS}_{fin}(A)$.

For two multisets m and m' we define the relation $m \geq m'$ and the addition $m + m'$ pointwise: $m \geq m'$ iff for each $a \in A$ holds $m[a] \geq m'[a]$ and $(m + m')[a] \mathrel{\widehat{=}} m[a] + m'[a]$. For $m \geq m'$ we define the difference $m - m'$ pointwise by $(m - m')[a] \mathrel{\widehat{=}} m[a] - m'[a]$.

Algebras An *algebra* $\mathcal{A} = (A, OP)$ consists of a countable *base set* A of objects and a finite set OP of total *operations* $f : A_1 \times \ldots \times A_n \to A_{n+1}$ for some $n \in \mathbb{N}$ where $A_i \subseteq A$.

Variables, terms and assignments Let X be a set of symbols[2] and $\mathcal{A} = (A, OP)$ an algebra. A pair $\mathcal{X} = (X, dom)$ is called *variable set* for $\mathcal{A}$ iff $dom : X \to 2^A$ is a mapping, that associates to each $x \in X$ a *sort* $dom(x) \subseteq A$. We define the set of terms with respect to a particular *sort* $A' \subseteq A$ of a given algebra: The set of *terms* over algebra $\mathcal{A}$ and a variable set $\mathcal{X} = (X, dom)$ is denoted by $\mathbf{T}_{\mathcal{A}}(\mathcal{X}, A')$ and inductively defined by:

1. $x \in \mathbf{T}_{\mathcal{A}}(\mathcal{X}, A')$ iff $dom(x) \subseteq A'$,
2. If $f : A_1 \times \ldots \times A_n \to A_{n+1}$ is an operation of $\mathcal{A}$, $A_{n+1} \subseteq A'$, and $u_1, \ldots, u_n \in \mathbf{T}_{\mathcal{A}}(\mathcal{X}, A_i)$, then $\mathsf{f}(u_1, \ldots, u_n) \in \mathbf{T}_{\mathcal{A}}(\mathcal{X}, A')$

Note that we distinguish here between an operation f and the corresponding *operation symbol* f by using a different type face. But we do not strictly keep up this distinction in the paper. The distinction is necessary for technical reasons in the following definition of $\overline{\beta}$, only.

Now we introduce assignments in order to evaluate terms. For a variable set $\mathcal{X} = (X, dom)$ a mapping $\beta : X \to A$ is an *assignment* for $\mathcal{X}$ in $\mathcal{A} = (A, OP)$, if $\beta(x) \in dom(x)$ for each $x \in X$. The set of all assignments for $\mathcal{X}$ in $\mathcal{A}$ is denoted by $\mathbf{ASS}(\mathcal{X}, A)$. An assignment $\beta : X \to A$ can be extended to the set of all terms $\overline{\beta} : \mathbf{T}_{\mathcal{A}}(\mathcal{X}, A) \to A$ inductively over the structure of terms:

1. for $x \in X$ let $\overline{\beta}(x) \mathrel{\widehat{=}} \beta(x)$
2. for $\mathsf{f}(u_1, \ldots, u_n) \in \mathbf{T}_{\mathcal{A}}(\mathcal{X}, A)$ let $\overline{\beta}(\mathsf{f}(u_1, \ldots, u_n)) \mathrel{\widehat{=}} f(\overline{\beta}(u_1), \ldots, \overline{\beta}(u_n))$

[2] We assume that X is disjoint from all other sets of symbols.

High-level Petri nets A (high-level) Petri net Σ consists of

- a *net* $N = (P, T; F)$, i.e. two disjoint and finite sets P and T of *places* and *transitions* related by a *flow relation* $F \subseteq (P \times T) \cup (T \times P)$. Elements of F are called *arcs*.
- an *algebra* $\mathcal{A} = (A, OP)$ and
- a distinguished *token domain* $D \subseteq A$ with $\mathbf{MS}_{fin}(D) \subseteq A$
- a *variable set* $\mathcal{X} = (X, dom)$ for $\mathcal{A}$,
- an *arc inscription* $i : F \to \mathbf{T}_{\mathcal{A}}(\mathcal{X}, \mathbf{MS}_{fin}(D))$
- the *initial state* $M_0 \in \mathcal{M}(P, D)$, where $\mathcal{M}(P, D) = \{M \mid M : P \to \mathbf{MS}_{fin}(D)\}$ is the set of all states of Σ.

The relation $\geq$ and the operations $+$ and $-$ can be extended to states place-wise. For $M, M' \in \mathcal{M}(P, D)$ we define: $M \geq M'$ iff for each $p \in P$ holds $M(p) \geq M'(p)$; $(M + M')(p) \mathrel{\widehat{=}} M(p) + M'(p)$. For $M \geq M'$ we define $M - M'$ by $(M - M')(p) \mathrel{\widehat{=}} M(p) - M'(p)$.

Dynamic behaviour of high-level nets Figure 5 illustrates an occurrence of a transition and the corresponding state change. In the left part of the figure, transition $t3$ is enabled in an occurrence mode, where agent a may receive a message from agent b as well as from i. The right figure shows the state after reception of the message from b. We introduce the

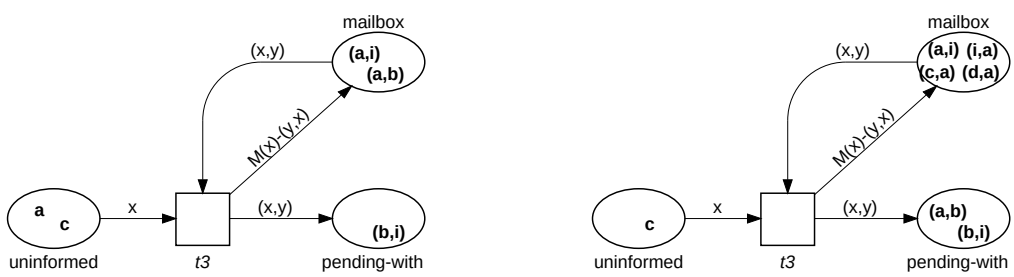

Figure 5: An occurrence of transition $t3$ in mode $[x = \mathsf{a}, y = \mathsf{b}]$

convention, that for $f \notin F : i(f) \mathrel{\widehat{=}} \mathcal{O}$. Now, for each transition $t \in T$ and each occurrence mode $\beta : X \to D$ we define the *enabling-condition* $t^-_\beta \in \mathcal{M}(P, D)$ and the *post-condition* $t^+_\beta \in \mathcal{M}(P, D)$ of transition t in mode β by $t^-_\beta(p) \mathrel{\widehat{=}} \overline{\beta}(i(p, t))$ and $t^+_\beta(p) \mathrel{\widehat{=}} \overline{\beta}(i(t, p))$ for each $p \in P$. Formally, both notions are states of the net.

A transition t is *enabled* in occurrence mode β at state M if the enabling condition holds, i.e. $M \geq t^-_\beta$. Then, t may *occur in mode* β resulting in a *follower state* $M' \mathrel{\widehat{=}} (M - t^-_\beta) + t^+_\beta$, written $M \longrightarrow M'$. Note, that in the follower state holds the post-condition $M' \geq t^+_\beta$.

A *run* of a Petri net is a (possibly infinite) sequence of states $\sigma = M_0, M_1, \ldots$ such that for each index n holds: $M_n \longrightarrow M_{n+1}$. If $\sigma = M_0 \ldots M_k$ is finite, then M_k does not enable any transition in any mode. Note, that each run starts with the initial state of the Petri net.

B: Propositions and Proof Techniques

In the sequel the semantics of propositions is defined and the classical P-invariant theory is formalized. Let $\Sigma = ((P, T; F), \mathcal{A}, D, \mathcal{X}, i, M_0)$ be a Petri net.

Temporal proposition Let φ be a state proposition. Then, $\Box\varphi$ and $\Diamond\varphi$ are called *temporal propositions*. A temporal proposition is valid for a Petri net iff it is valid for each run of the Petri net. A temporal proposition $\Box\varphi$ is valid for a run iff φ holds in each state M_i of σ. A temporal proposition $\Diamond\varphi$ is valid for a run iff there exist a state M_j in σ where φ holds.

State expressions A *state expression* $\gamma \in \mathbf{T}_{\mathcal{A}}((P, dom), B)$ is a term that uses as variables the places of the net where $dom(p) = \mathbf{MS}_{fin}(D)$ for each $p \in P$ where $B \subseteq A$ is some set, called the *type* of γ. The *value* of γ in some state $M \in \mathcal{M}(P, D)$ is denoted by $\gamma(M)$ and defined by $\gamma(M) \mathrel{\widehat{=}} \overline{\beta}_M(\gamma)$ where $\beta_M \in \mathbf{ASS}(P, \mathbf{MS}_{fin}(D))$ is defined by $\beta_M(p) \mathrel{\widehat{=}} M(p)$ for each $p \in P$. A state expression is called *linear*[3] iff for all $M_1, M_2 \in \mathcal{M}(P, D) :\ \gamma(M_1 + M_2) = \gamma(M_1) + \gamma(M_2)$. A linear state expression is called a P-invariant of Σ if its type is $\mathbf{MS}_{fin}(D)$ and iff for all $t \in T$ and all $\beta \in \mathbf{ASS}(\mathcal{X}, D) :\ \gamma(t^-_\beta) = \gamma(t^+_\beta)$.

Theorem: If γ is a P-invariant for Σ, then $\Box\gamma = \gamma(M_0)$ is a valid proposition for Σ.

A linear state expression is called *descending variant* for Σ if its type is $\mathbb{N}$ and iff for each $t \in T$ and each $\beta \in \mathbf{ASS}(\mathcal{X}, D)$ holds $\gamma(t^-_\beta) > \gamma(t^+_\beta)$.

Theorem: If γ is a descending variant for Σ, then $\Diamond\mathsf{lockout}_\Sigma$ is a valid proposition for Σ, where $\mathsf{lockout}_\Sigma$ denotes the state predicate valid in all the states that do not enable any transition of Σ.

Finally, we introduce an equivalent characterization of a P-invariant, which allows to check a P-invariant by checking the validity of one equation for each transition. Let $t \in T$ be a transition of the net. With $t^-, t^+ : P \to \mathbf{T}_{\mathcal{A}}(\mathcal{X}, \mathbf{MS}_{fin}(D))$ we denote the following *substitutions*:

$$t^-(p) \mathrel{\widehat{=}} i(p, t) \quad \text{and} \quad t^+(p) \mathrel{\widehat{=}} i(t, p)$$

The application of these substitutions to a linear state expression γ is straight-forward. We denote these applications with $t^-(\gamma)$ and $t^+(\gamma)$.

Proposition A linear state expression γ is a P-invariant of Σ iff for each transition t of Σ the equation $t^-(\gamma) = t^+(\gamma)$ is valid in algebra $\mathcal{A}$.

Similarly, we can characterize descending variants as follows:

Proposition A linear state expression $\gamma \in \mathbf{T}_{\mathcal{A}}(P, \mathbb{N})$ is a *descending variant* of Σ, iff for each transition t of Σ the in-equation $t^-(\gamma) > t^+(\gamma)$ holds in the algebra $\mathcal{A}$.

[3] All state expressions used in this paper are linear by construction, but we do not elaborate on that here.