Distributed Algorithms for Networks of Agents

W. Reisig, E. Kindler, T. Vesper, H. Völzer, and R. Walter

Humboldt - Universität zu Berlin

Abstract A new kind of algorithms, called *distributed algorithms*, has emerged during the last decade, aimed at efficiently solving problems that occur whenever distributed computing systems are to be made applicable to real-world problems.
Distributed computing systems are frequently organized as *networks of agents*, with each agent asynchronously interacting with some of its neighboring agents. Algorithms running on such networks are called *distributed*.
A *network algorithm* is a *schema*, intended to run on any network in a whole class of networks. Such an algorithm can be modeled as a *high-level Petri net schema*. Each interpretation of the schema yields an algorithm for a concrete network.
This paper suggests a variety of Petri net models of network algorithms, formally represents their most decisive properties, and proves their validity. To this end, well-known techniques such as place invariants and traps are adjusted to Petri net schemata, and new techniques to prove *progress* properties are suggested.

Introduction

The paradigm of computing is shifting away from centralized *one agent systems* towards decentral *networks of agents*. Each agent may exchange messages with neighboring agents in the network.

Agents may jointly solve any kind of problems, frequently initiated by one of the agents. In most cases, all but the initiator agent are running identical algorithms. Each agent is usually aware of its neighboring agents only; thus no agent controls the entire network. An algorithm of this kind will be called a *network algorithm* in the following.

A network algorithm is not intended to run on just one fixed network. Rather, a network algorithm is a *schema* of algorithms, which run on any network in a whole class of networks, such as the connected networks, the ring- or tree-shaped networks, etc.

A Petri net model of a network algorithm must reflect this aspect. Consequently, a network algorithm will be modeled by Petri net *schema*. A Petri net schema in particular includes symbols to denote sets and functions. Any instantiation of those functions turns the schema into a concrete high-level Petri net, representing an algorithm on a concrete network.

This contribution assumes basic knowledge of high-level Petri nets. Some few, fairly obvious new concepts will be employed, introduced in an intuitive way by

help of the considered case studies. This applies likewise to verification techniques: We employ well established algebraic place invariants as well as newly designed *weighted traps* and *pick-up rules* for progress properties. The considered case studies will clarify when and how they are to be used. In fact, choice and order of the case studies are governed by increasingly involved analysis techniques.

1 Consensus in Networks

1.1 The problem

A consensus algorithm organizes consensus about some contract or agreement among the agents of a network. This is not trivial in case all agents are homogeneous, each agent can exchange messages with some neighbors only and there is no other communication medium available, e.g., a broker or mediator who could communicate with each agent.

An algorithm will be constructed in the following, to solve this problem for any network of agents. The central activity of each agent is broadcast and receipt of messages, containing proposals for a joint contract. Each agent u is assigned a fixed set of other agents which u is to communicate with. Upon receiving a message, an agent returns a receipt to its sender. The algorithm does not guarantee that consensus will ever be reached. But consensus will turn out to be *stable*: Once reached, it remains.

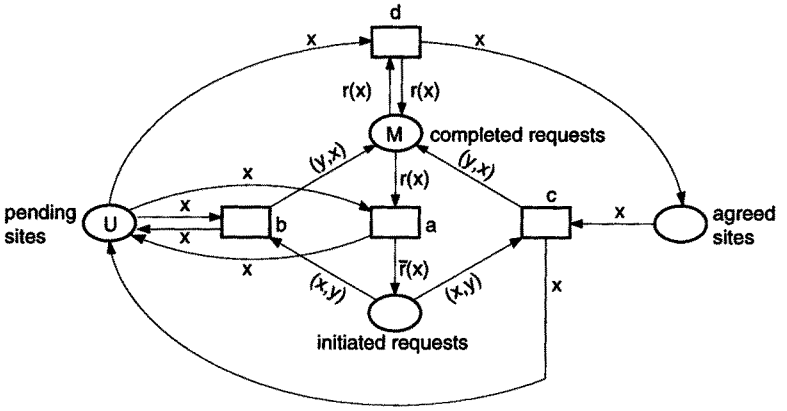
1.2 The algorithm

Figure 1.1 shows a Petri net schema representation, $\Sigma_{1.1}$, of the consensus algorithm.

U and M are symbols, to be instantiated by a set and a relation, as the text in the figure's lower part explains. U represents the set of *agents*, and M the relation of neighborhood, with $M(x) = \{y|(x,y) \in M\}$ denoting the set of sites which u is to send messages to. Notice that M is not required to be symmetric. As a general rule, a message is always represented as a pair (*receiver*, *sender*). The equations in the text of Fig. 1.1 hence specify $r(x)$ and $\bar{r}(x)$ as the set of all messages to be received or to be sent, respectively, by x.

Initially, each agent is pending and each request is completed. In this situation, an agent u may send each of its neighbors v a message (transition a in mode $x = u$). Upon receiving a message (u, v) from v, a *pending* agent u returns a receipt, (v, u), to the message's sender v (transition b in mode $x = u$, $y = v$). A *pending* agent u may turn *agreed*, provided all its messages are completed (transition d in mode $x = u$). Finally, upon receiving a message, an *agreed* agent u turns *pending* (transition c in mode $x = u$).

Obviously, at any time, an agent is either *pending* or *agreed*, and a message is either *completed* or *initiated*. The algorithm does not guarantee that the sites eventually all will *agree*. However, the algorithm guarantees *stability*: If all sites do *agree*, no site will return to *pending*; the algorithm *terminates* in this case.



sort	site	fct	r, r̄ :site → set of messages
sort	message = site × site	var	x, y : site
const	U : set of sites		r(x) = {x} × M(x)
const	M : set of messages		r̄(x) = M(x) ×{x}

Figure 1.1. Basic algorithm for distributed consensus

1.3 Algebraic place invariants

Here we are interested in techniques to prove the above-mentioned *stability* of the consensus algorithm. Of course, it is not possible to just inspect all reachable states with all agents *agreed*, because the net $\Sigma_{1.1}$ is just a schema for in fact infinitely many *models*. Hence we look for techniques that can be applied to the syntactical representation of $\Sigma_{1.1}$ and would allow to express and prove properties that hold in *all* models. One technique of this kind are the well-known *algebraic place invariants*. In fact, they support proof of stability, but they are not sufficient. In addition, *symbolic traps* will be used.

Both, place invariants and symbolic traps, employ syntactical terms which at any concrete interpretation represent *linear functions*. For technical details, we refer to the Appendix. Figure 1.2 shows the matrix, initial state and two place invariants of the algorithm of $\Sigma_{1.1}$. Shorthands for places, as introduced in Fig. 1.2, will be applied throughout this chapter.

As usual, the arc inscriptions are taken as matrix entries, with the minus symbol representing arcs from places to transitions, and $\tau - \tau = 0$ for all terms τ (the term 0 is usually skipped). Each invariant entry is a term, including at most one variable; for convenience, the name of the corresponding place serves this purpose. In Fig. 1.2, the terms *are* the corresponding variables, up to the term $\bar{D}$. This term, with variable D ranging over relations, denotes $\{(v,u)|(u,v) \in D\}$, i.e. inverts the pairs at D. The product of a matrix entry τ with an invariant entry σ is gained by substitution of τ into each occurrence of the unique variable in σ. For example, $-r(x) \cdot C = -r(x)$, and $\bar{r}(x) \cdot \bar{D} = \overline{(\bar{r}(x))}$. The inner product

$\Sigma_{1.1}$	a	b	c	d	s_0	i_1	i_2
A			x	$-x$	U	A	
B			$-x$	x		B	
C	$-r(x)$	(y,x)	(y,x)		M		C
D	$\bar{r}(x)$	$-(x,y)$	$-(x,y)$				$\bar{D}$

A: pending agents C: acknowledged messages
B: agreed agents D: initiated messages

Figure 1.2. Matrix, initial state s_0, and two place invariants, i_1 and i_2, to the consensus algorithm, $\Sigma_{1.1}$

of the column a with i_2 then is $\underline{a} \cdot i_2 = -r(x) \cdot C + \bar{r}(x) \cdot \bar{D} = -r(x) + \overline{(\bar{r}(x))} = -r(x) + r(x) = 0$, because $(v,u) \in r(u)$ iff $(u,v) \in \bar{r}(u)$, according to Fig. 1.1. Likewise, $\underline{b} \cdot i_2 = (y,x) \cdot C - (x,y) \cdot \bar{D} = (y,x) - (x,y) = (y,x) - (y,x) = 0$. The product of each matrix column with each of i_1 and i_2 evaluate to 0, hence both i_1 and i_2 are in fact place invariants of $\Sigma_{1.1}$. Furthermore, $s_0 \cdot i_1 = U \cdot A = U$, hence for each reachable state s holds $s(A) + s(B) = U$; this will for short be written

$$A + B = U. \tag{1}$$

Likewise holds: $s_0 \cdot i_2 = M \cdot C = M$, hence the equation

$$C + \bar{D} = M \tag{2}$$

holds at each reachable state.

1.4 Symbolic traps

A further property will be required, that follows from an *initialized symbolic trap* of $\Sigma_{1.1}$. A trap consists of a set P of places and expressions I^p for each $p \in P$, such that p is the only variable of I^p, and for each transition occurrence t holds: If t removes the set q_0 of tokens from P and adds the set q_1 of tokens to P, then

$$\bigcup_{p \in P} I^p(q_0) \subseteq \bigcup_{p \in P} I^p(q_1). \tag{3}$$

For obvious reasons, P is called the *domain*, and the expressions I^p are called the *weight functions* of the trap. For example, $P = \{A, C\}$, $I^A = r(A)$, and $I^C = C$ form a trap of Σ: Transitions a and b retain the token load on $I(A) \cup C$, transition c adds tokens; d is the only nontrivial transition. d removes x from A and $r(x)$ from C, hence d removes $r(x)$ from both $r(A)$ and C, but d returns $r(x)$ to C, hence d meets requirement (3).

The *initial value* of a trap is the union of the weighted tokens that initially occur in its domain. For example, the initial value of the above described trap

of $\Sigma_{1.1}$, with domain $\{A, C\}$ and weight functions I^A and I^C, is $I^A(s_0(A)) \cup I^C(s_0(C)) = I^A(U) \cup I^C(\emptyset) = r(U) \cup \emptyset = M$. The initial value V of a trap with domain $P = \{P_1, \ldots, P_n\}$ and weight functions $I^{P_1}, \ldots, I^{P_n}$ yields the inequality

$$I^{P_1}(s_0(p_1)) + \cdots + I^{P_n}(s_0(p_n)) \geq V, \tag{4}$$

which holds for each reachable state.

For the above example we obtain this way that

$$r(A) + C \geq M \tag{5}$$

holds at each reachable state of $\Sigma_{1.1}$.

1.5 Proving stability

Stability of $\Sigma_{1.1}$, as informally described at the end of Sect. 1.2, can now formally be represented by the formula

$$B = U \rightarrow A = \emptyset \wedge D = \emptyset, \tag{6}$$

claimed to hold at each reachable state: $A = \emptyset \wedge D = \emptyset$ implies that no transition is enabled. With $A = \emptyset$, transitions a, b, and d are not enabled. $D = \emptyset$ likewise implies that c is not enabled. Hence, in fact $A = \emptyset \wedge D = \emptyset$ implies stability.

The above equations (1) and (2) together with the inequality (5) suffice to prove (6): Equation (1) implies $r(A) + r(B) = r(U)$; hence, with $r(U) = M$ following from the specifications of $\Sigma_{1.1}$, we obtain

$$r(A) + r(B) = M. \tag{7}$$

Subtraction of (5) from the sum of (2) and (7) yields

$$r(B) + \bar{D} \leq M. \tag{8}$$

Furthermore, $B = U \rightarrow r(B) = M$, and $r(B) = M \rightarrow \bar{D} = \emptyset$ (by (8)), and $\bar{D} = \emptyset \rightarrow D = \emptyset$. Transitivity of implication now yields

$$B = U \rightarrow D = \emptyset. \tag{9}$$

Furthermore, $B = U \rightarrow A = \emptyset$ follows from (1), hence (6).

This proof shows stability for *each* network N of agents and each reachable state of N. It is exclusively based on the syntactical units of Fig. 1.1, and on theorems about Petri net schemata.

2 Phase Synchronization

2.1 The problem

Network algorithms work frequently in *rounds* or *phases*: Each agent eventually returns to its initial state, thus entering its next phase.

A synchronization mechanism is occasionally required, that guarantees synchronized execution of phases: No agent begins its $(k+1)$st phase unless all agents have completed their k-th phase. Stated differently, whenever two agents are busy at the same time, they are executing the same phase.

It is not entirely trivial to organize this kind of behavior in a network of agents that can exchange messages with some neighbors only, lacking any global agent such as a mediator, who could communicate with each agent.

A phase synchronization algorithm will be presented in the following, to run on any connected, acyclic network (undirected tree). Figure 2.1 shows an example. Its *leaves* are a, c, g, h, j, k. Adding or deleting an arc without adding

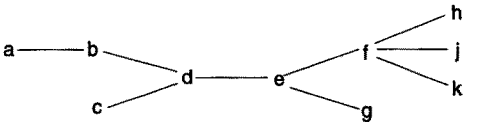


Figure 2.1.

or deleting nodes would make the network cyclic or unconnected, respectively.

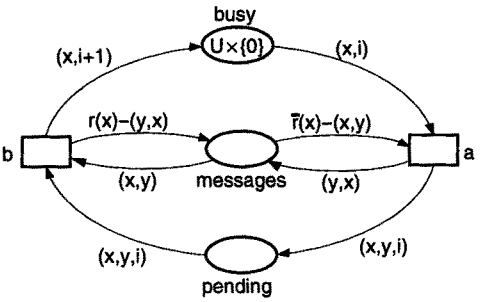
2.2 The algorithm

Figure 2.2 shows the phase synchronization algorithm, $\Sigma_{2.2}$. Each agent alternates between the states *busy* and *pending*. The agent's round number increases by 1 upon reaching *busy*.

Whenever changing its actual state, an agent consumes and produces *messages* from and for neighboring agents, respectively. A message is represented as (*receiver*, *sender*). (The multiset notation $\bar{r}(x) - (x, y)$ denotes conventionally $\bar{r}(x) \setminus \{(x, y)\}$ and is defined only if $(x, y) \in \bar{r}(x)$.)

All agents are initially busy in their 0-th round, and no message is available. For an agent u, occurrence of transition a in a mode $x = u$ and $i = 0$ furthermore requires the set $\bar{r}(u) \setminus \{(u, v)\}$ of messages, for some $v \in W(u)$. With no messages for u available, this set must be empty, hence $\bar{r}(u) = \{(u, v)\}$, hence v must be the only neighbor of u. This, in fact, applies to the leaves of the network, viz. a, c, g, h, j, k in Fig. 2.1. Occurrence of transition a for some agent u yields a message (v, u) to u's unique neighbor, v. Some of the inner agents may then enable a (in Fig. 2.1, these agents are b and f). All agents are eventually pending and two messages, formed (u, v) and (v, u) of neighboring agents u and v are available. In Fig. 2.1, u and v may be d and e, respectively. But any other neighboring agents may likewise play this role. For example, agent h in Fig. 2.1 may remain busy until all other agents are pending. This situation retains one message, (h, f). Move of h to *pending* then adds the message (f, h).

Messages formed (u, v) and (v, u) start the wave back to *busy*. The partial order of occurrences of transition a is now reversed for transition b: The last



sort site
sort message = site × site
const U: set of sites
const W : set of (sites × sites)
fct r, r̄ : site → set of messages
var x, y : site
var i : nat

$W = W^{-1}$
$x,y \in U \rightarrow x\,W^{*}\,y$
$W_1 = U$
$x_0 W\, x_1 \ldots x_n W\, x_{n+1} \wedge$
$x_{i-1} \neq x_{i+1}$ for i=1,...,n
$\rightarrow x_0 \neq x_n$
$r(x) = W(x) \times \{x\}$
$\bar{r}(x) = r(x)^{-1}$

Figure 2.2. Phase synchronization

agents having reached *pending* will be the first ones to go *busy* in the next round. The last agents to go *busy* again are the leaves.

A "lazy" site u may still be *pending* with a message (u, v) in round i, while its "diligent" neighbor v, in its $(i+1)$st round has sent a further message to u. Hence two messages formed (u, v) may coincidentally be at place *messages*. This does not perfectly meet the formalism of the Appendix, which disallows more than one copy of a token. To fix this problem, either include the round number as a further component to each message, or canonically extend the formal model to cover multiple occurrences of tokens, as suggested in e.g. [Weber et al 98].

2.3 Properties to be proven

Two properties are to be proven. Firstly, two *busy* agents are in the same round. As a shorthand, for a place p and a token a, the term $p.a$ denotes at a given state that there is at least one copy of the token a at the place p. Hence we have to show that the formula

$$busy.(u, n) \wedge busy.(v, m) \rightarrow n = m \tag{10}$$

holds at each reachable state of $\Sigma_{2.2}$. In the framework of temporal logic, this is a typical *safety property*, stating that "never something bad happens".

The second property to be proven states that each agent *will* eventually reach each round. More formally: For each interleaved run $s_0 \xrightarrow{t_1} s_1 \xrightarrow{t_2} \ldots$ of $\Sigma_{2.2}$ holds:

If at state s_w holds $busy.(u,n)$ then there exists an index $j \geq w$ such that at state s_j holds $busy.(u,n+1)$. We denote this by

$$busy.(u,n) \mapsto busy.(u,n+1), \tag{11}$$

adapting notational conventions from temporal logic, particularly from [Chandy, Misra 88]. There, $\mapsto$ is called *leads-to*.

This kind of properties has rarely been considered in the framework of Petri nets. In temporal logic, (11) is a typical *liveness property*, stating that "eventually something good will happen". In particular, this kind of liveness properties is entirely different from well-established *reachability*, which just claims the *chance* to reach a distinguished state. Liveness, to the contrary, states that a distinguished (kind of) state will *inevitably* be reached in each run (we always assume *maximal* runs; i.e. runs which are infinite or terminate with no transition enabled).

2.4 Place invariants

As a matter of convenience we employ shorthands of pairs and triples:
$(a,b)_1 = (a,b,c)_1 = a$,
$(a,b,c)_{1,2} = (a,c,b)_{1,3} = (b,a,c)_{2,1} = (a,b)$,
$\overline{(a,b)} = (b,a)$
which lift canonically to (binary or ternary) relations.

Figure 2.3 shows three place invariants to the phase synchronization algorithm, $\Sigma_{2.2}$. i_1 is quite obvious, whereas i_2 and i_3 are more involved.

	a	b	s_0	i_1	i_2	i_3
A	$-(x,i)$	$(x,i+1)$	$U \times \{0\}$	A_1		$\alpha(A) - \bar{\alpha}(A)$
B	$-\bar{r}(x)$	$r(x)$			$B + \bar{B}$	$\bar{B} - B$
	$+(x,y)$	$-(y,x)$				
	$+(y,x)$	$-(x,y)$				
C	(x,y,i)	$-(x,y,i)$		C_1	$r(C_1) + \bar{r}(C_1)$ $-2C_{1,2} - 2C_{2,1}$	$\beta(C_{1,3}) - \bar{\beta}(C_{1,3})$

A:busy
B:messages
C:pending

$\alpha(u,n) := 2n \cdot r(u)$
$\bar{\alpha}(u,n) := 2n \cdot \bar{r}(u)$
$\beta(u,n) := (2n+1) \cdot r(u)$
$\bar{\beta}(u,n) := (2n+1) \cdot \bar{r}(u)$

Figure 2.3. Matrix, initial state, and four place invariants to $\Sigma_{2.2}$

$\Sigma_{2.2}$ has three important place invariants. Two of them are quite intuitive. First of all, $A_1 + C_1 = U$, which for each $u \in U$ implies

$$A_1.u + C_1.u = 1 \ . \tag{12}$$

Hence each site is always either busy or pending. Furthermore, i_1 implies

$$|C_1| = |C_{1,2}|, \tag{13}$$

hence each site u has always a unique round number, and if pending, it is pending with a unique site v.

The place invariant $B + \overline{B} + r(C_1) + \overline{r}(C_1) = 2(C_{1,2} + C_{2,1})$ relates pending neighbors to their mutual messages. For each pair (u, v) of neighboring sites this implies

$$\begin{aligned} &B.(u,v) + B.(v,u) + r(C_1).(u,v) + r(C_1).(v,u) \\ &= 2 \cdot C_{1,2}.(u,v) + 2 \cdot C_{1,2}.(v,u). \end{aligned} \tag{14}$$

Furthermore, $\neg C_1.u \wedge C_1.v$ implies $r(C_1).(v,u) = C_{1,2}.(u,v) = 0 \wedge r(C_1).(u,v) = C_{1,2}.(v,u) = 1$, hence, by (14), $B.(u,v) + B.(v,u) = 1$, hence with (12),

$$A_1.u \wedge \neg A_1.v \rightarrow B.(u,v) \vee B.(v,u) \ . \tag{15}$$

The place invariant above furthermore implies

$$|B| + |\overline{B}| = 2|C_{1,2} + C_{2,1}| - |r(C_1)| - |\overline{r}(C_1)| \ . \tag{16}$$

The third place invariant is $\alpha(A) + \overline{B} + \beta(C_{1,3}) = \overline{\alpha}(A) + B + \overline{\beta}(C_{1,3})$, which implies for all $u, v \in U$:

$$\alpha(A).(u,v) + B.(v,u) + \beta(C_{1,3}).(u,v) = \alpha(A).(v,u) + B.(u,v) + \beta(C_{1,3}).(v,u) \ . \tag{17}$$

This invariant links all places of $\Sigma_{2.2}$.

2.5 Busy neighbors don't exchange messages

In case two neighboring sites u and v are both busy, there is no message available from u to v or from v to u. In terms of $\Sigma_{2.2}$ this reads for all $u \in U$ and $v \in W(u)$:

$$A_1.u \wedge A_1.v \rightarrow B.(u,v) = B.(v,u) = 0 \ . \tag{18}$$

Upon proving (18), assume a state s with $s \models A_1.u \wedge A_1.v$. Then at s holds $A_1.u = A_1.v = 1$, hence $C_1.u = C_1.v = 0$ (by (12)), hence $C_{1,2}.(u,v) = C_{1,2}.(v,u) = 0$ (by (13)), hence the proposition, by (14). □

2.6 A property of neighboring pending sites

A neighbor v of a pending site u is pending with u, or u is pending with v. In terms of $\Sigma_{2.2}$, for $u \in U$ and $v \in W(u)$,

$$C_1.u \to C_{1,2}.(u,v) \vee C_{1,2}.(v,u) \ . \tag{19}$$

Proof of (19) assumes a state s with $s \models C_1.u = 1$. Then at s holds for all $w \in W(u)$: $r(C_1).(u,w) = 1$, hence particularly $r(C_1).(u,v) = 1$, hence $C_{1,2}.(u,v) + C_{1,2}.(v,u) \geq 1$, by (14), hence the proposition. □

2.7 A site is pending with a busy neighbor

A pending site v with a busy neighbor u is pending with u. (Hence, with (13), at most one neighbor of a pending site is busy). In terms of $\Sigma_{2.2}$, for $u \in U$ and $v \in W(u)$,

$$A_1.u \wedge C_1.v \to C_{1,2}.(v,u) \ . \tag{20}$$

Proof of (20) combines two properties of $\Sigma_{2.2}$: First, $C_1.v$ implies $C_{1,2}.(u,v) \vee C_{1,2}.(v,u)$ by (19). Second, $A_1.u$ implies $\neg C_1.u$ by (12), hence $\neg C_{1,2}.(u,v)$. □

2.8 Three pending neighbors form a sequence

Assume a site v, pending with w. Then each other pending neighbor u of v is pending with v. In $\Sigma_{2.2}$ this reads for $v \in U$ and $u, w \in W(v)$:

$$C_1.u \wedge C_{1,2}.(v,w) \to C_{1,2}.(u,v) \ . \tag{21}$$

Proof of (21) combines two properties of $\Sigma_{2.2}$: First, $C_1.u$ implies $C_{1,2}.(u,v) \vee C_{1,2}.(v,u)$ by (19). Second, $C_{1,2}.(v,w)$ implies $\neg C_{1,2}.(v,u)$, by (12). □

2.9 Busy neighbors are in the same round

If two neighbors u and v are both busy, they operate in the same round. In $\Sigma_{2.2}$ this reads for $u \in U$, $v \in W(u)$, and $n, m \in \mathbb{N}$:

$$A.(u,n) \wedge A.(v,m) \to n = m \ . \tag{22}$$

To prove (22), let s be a reachable state of $\Sigma_{2.2}$ with $s \models A_1.u \wedge A_1.v$. Then at s holds $C_1.u = C_1.v = 0$ by (12), hence $\beta(C_{1,3}).(u,v) = \beta(C_{1,3}).(v,u) = 0$. Furthermore, $B.(u,v) = B.(v,u) = 0$, by (18). Combining both properties, (17) yields $\alpha(A).(u,v) = \alpha(A).(v,u)$. Then for each $n \in \mathbb{N}$, $A.(u,n) \to A.(v,n)$. Then (22) follows with (12). □

2.10 A property of chains

Given $u_0, \ldots, u_n \in U$, the sequence $u_0 \ldots u_n$ is a *chain* if $u_{i-1} \in W(u_i)$ for $i = 1, \ldots, n$, and $u_{i-1} \neq u_{i+1}$ for $i = 1, \ldots, n-1$.

Assume a chain $u_0 \ldots u_n$, starting with a busy site, u_0, followed by a pending site, u_1. Then all follower sites $u_2, \ldots, u_n$ are pending. In $\Sigma_{2.2}$ this reads

$$A_1.u_0 \wedge C_1.u_1 \rightarrow C_1.u_i \text{ for all } i = 1, \ldots, n \ . \tag{23}$$

To prove (23), let s be a reachable state with $s \models A_1.u_0 \wedge C_1.u_1$. Then at s holds $C_{1,2}.(u_1, u_0)$ by (20). Then

$$\neg C_{1,2}.(u_1, u_2) \tag{24}$$

by (12). Now, contradicting (23), assume an index $1 \leq i \leq n$ with $s \models \neg C_1.u_i$. Let j be the smallest of those indices. Then at s holds $A_1.u_j$ by (12), hence $C_{1,2}.(u_{j-1}, u_j)$, by (20). Then $C_{1,2}.(u_{i-1}, u_i)$ for $i = 2, \ldots, n$ by iterated application of (21). Then in particular $C_{1,2}.(u_1, u_2)$, which contradicts (24). □

2.11 Proof of the state property (10)

We are now prepared to prove (10) as follows:

Let s be a reachable state with $s \models A.(u, n) \wedge A.(v, m)$. Then there exists a chain $u_0 \ldots u_n$ in U with $u_0 = u$ and $u_n = v$. Then $s \models A_1.u_i$ for $i = 0, \ldots, n$, by (23) and (12). Then at s holds $A.(u_i, n)$ for $i = 0, \ldots, n$ by iteration of (22). Hence $n = m$.

2.12 Pending sites have pending messages

Here we start proof of the liveness property (11). First, we observe pending messages in case all sites are pending:

$$C_1.U \rightarrow |B| > 0 \ . \tag{25}$$

Proof of (25) is based on the observation that an undirected tree with n nodes has $n - 1$ arcs. Hence, in $\Sigma_{2.2}$,

$$|r(U)| = |\overline{r}(U)| = 2|U| - 2 \ . \tag{26}$$

Then $C_1.U \rightarrow |B| + |\overline{B}| = 2|C_{1,2} + C_{2,1}| - |r(C_1)| - |\overline{r}(C_1)|$ (by (16)) $= 4|U| - 2(2|U| - 2)$ (by (26)) $= 4$.

2.13 $\Sigma_{2.2}$ is deadlock free

Each reachable state of $\Sigma_{2.2}$ enables at least one action. (27)

Proof. Let s be a reachable state of $\Sigma_{2.2}$. 1st case: $s \models A_1.u$ for at least one $u \in U$. Then there exists a chain $u_0 \ldots u_n$, $n \geq 0$, of sites with $s \models A_1.u_i$ for all $i = 0, \ldots, n$, and $\neg A.v$ for all $v \in W(u_n) - u_{n-1}$. Hence for all such v holds $s \models B.(v,u) \vee B.(u,v)$, by (15). Now we distinguish two cases: Firstly, $s \models B.(u,v)$ for all $v \in W(u_n) - u_{n-1}$. Then s enables $a(u_n, u_{n-1}, k)$, where $s \models A.(u_n, k)$. Otherwise, there exists some $v \in r(u_n) - u_{n-1}$ with $s \models B.(v,u)$. Furthermore, $s \models C.(v,u,k)$ for some $k \in \mathbb{N}$ (with (12)). Then s enables $b(v,u,k)$. 2nd case: There is no $u \in U$ with $s \models A_1.u$. Then $s \models C_1.U$ (with (12)). Then $|B| > 0$, by (25). Hence there exist $u, v \in U$ with $s \models B.(u,v)$. Then $s \models C.(u,v,k)$ for some $k \in \mathbb{N}$, by (14). Then s enables $b(u,v,k)$. □

2.14 The weight function γ

A function $\gamma(u,v)$ will be considered, which for neighbors u and v yields an integer value $\gamma(u,v)$ at any given state s. Values $\gamma(u_{i-1}, u_i)$ remain in a limited interval for all chains $u_0 \ldots u_n$, and occurrences of transitions increase those values. For $u, v \in U$, let

$$\gamma(u,v) := B.(v,u) + \Sigma_{n \in \mathbb{N}}(2n \cdot A.(u,n) + (2n+1) \cdot C_{1,3}.(u,n)) \ . \tag{28}$$

Then (27) implies

$$\gamma(u,v) = \gamma(v,u) \ . \tag{29}$$

Furthermore, for neighbors w of u, $C_{1,2}.(u,w) = r(C_1).(u,w)$; hence $B.(w,u) \leq 2$ (by (14)), hence

$$|\gamma(u,v) - \gamma(u,w)| \leq 2 \tag{30}$$

again by (14). Then for each sequence $u_0 \ldots u_k$ of sites, (29) and (30) imply

$$|\gamma(u_0, u_1) - \gamma(u_{k-1}, u_k)| \leq 2(k-1) \ . \tag{31}$$

2.15 Proof of the liveness property (11)

Inspection of $\Sigma_{2.2}$ yields for each step $r \xrightarrow{t} s$ with $t = a(u,v,i)$ or $t = b(u,v,i)$:

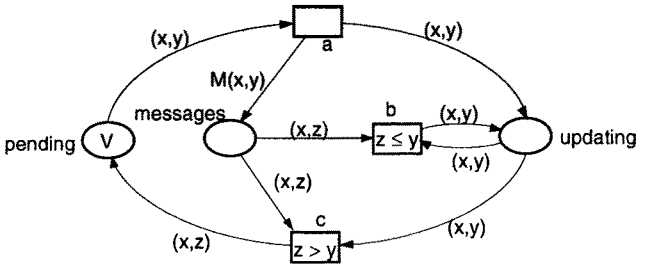
$$\text{If } \gamma(u,v) = n \text{ at state } r, \text{ then } \gamma(u,v) > n \text{ at state } s \ . \tag{32}$$

Property (27) implies at least one pair (u,v) of neighbors with infinitely many occurrences of $a(u,v,i)$ and $b(u,v,i)$. Then in the set of all reachable states, $\gamma(u,v)$ is not limited, by (32). This applies to all neighbors u, v, by (31). Hence $A.(u,n) \mapsto A.(u,n+1)$.

3 Leader Election and Spanning Trees

3.1 A leader election algorithm

The sites of a network are frequently supposed to elect one site as their *leader*. In case the leader site crashes, a new leader must be elected. The sites are given unique names to this end (e.g., integer numbers) and a total order is assumed on those names.



sort	site	var	x, y, z : site
sort	state : site × site		x,y ∈ U → x W* y
const	U : set of sites		$W_1 \cup W_2 = U$
const	V,W : set of states		V = {(u,u) | u ∈ U}
	≤ : total order on U		M(x,y) = W(x) × {y}
fct	M : state → set of states		

Figure 3.1. Basic leader election

Figure 3.1 gives a distributed algorithm for the election of a leader in any connected network. Initially, each site is *pending* and assumes its own name as a candidate for the leader. In later states, a *pending* site holds a better candidate, i.e., one with a larger name. Generally, a *pending* site u together with its actual candidate v is represented as a state (u, v). Upon *pending* with v, u informs each neighbor in $W(u)$ about v by action $a(u, v)$ and then becomes *updating*. An *updating* site u with its actual leader candidate v may receive a *message* (u, w). In case the newly suggested candidate, w, does not exceed v, the site u remains *updating* with v (action $b(u, v, w)$). Otherwise u goes *pending* with the new candidate w (action $c(u, v, w)$) and continues as described above.

A message $(w, v) \in M(u, v)$ takes the form of a state, with u informing the site w about v as a candidate for the leader. There may occur multiple copies of identical messages (as in case of communication protocols). This can easily be fixed, by extending each message with its sender.

Given a connected network with a finite set U of sites and a total order $\leq$ on U, the algorithm terminates with *updating* all pairs (u, w), where $u \in U$ and w is the maximal element of U.

Two neighbors u_0, u_1 of a site w may both be *pending* with the same candidate, v. Concurrent occurrences of $a(u_0, v)$ and $a(u_1, v)$ then yield two identical messages (w, v). This does not perfectly meet the formalism of the Appendix, which disallows more than one copy of a token. To fix this problem, either include the sender as a further component to each message, or canonically extend the formal model to cover multiple occurrences of tokens, as suggested in e.g. [Weber et al 98].

3.2 Property to be proven

The crucial property to be proven is a typical liveness property (in the temporal logic framework, c.f. Sect. 2.3): Each run terminates with each agent being *informed* about the leader's number. Using the *leads-to* operator already used in (11), the initial state s_Σ of $\Sigma_{3.1}$, the maximal agent *max* and the formula

$$\pi = \text{updating}.U \times \{\text{max}\} \wedge \text{pending} = \emptyset \wedge \text{messages} = \emptyset \tag{33}$$

we have to show

$$s_{\Sigma_{3.1}} \mapsto \pi. \tag{34}$$

As explained in Sect. 2.3, (34) states that in each interleaved run $s_0 \stackrel{t_1}{\to} s_1 \stackrel{t_1}{\to} \ldots$, each occurrence s_k of s_Σ is followed by a state s_{k+i} at which π holds.

Proof of (34) can considerably be eased by help of *concurrent runs*. This notion will be considered in the following.

3.3 Concurrent runs

In its essence, a concurrent run consists of the transition occurrences of an interleaved run, partially ordered by their causal dependencies. As an example, Fig. 3.2 shows a network of three agents, $U = \{1, 2, 3\}$. Arrows indicate the

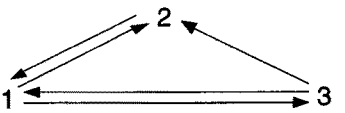


Figure 3.2.

neighboring relation. Representing each occurrence of a with valuation $x = u$ and $y = v$ by auv and each occurrence of b or c with valuation $x = u, y = v$ and $z = w$ by $buvw$ and $cuvw$, respectively, one of the interleaved runs of the instantiation of $\Sigma_{3.1}$ by the above network is

$$\begin{aligned} &a11\text{-}a22\text{-}a33\text{-}c113\text{-}b221\text{-}b331\text{-}a13\text{-} \\ &b132\text{-}c223\text{-}a23\text{-}b133\text{-}b233\text{-}b333. \end{aligned} \tag{35}$$

It describes the initial occurrences of transition a for all three agents, followed by agent 1's adoption of the better candidate, 3, and the other two agents' deletion of agent 1 as a candidate for the leader. Then agent 1 suggests 3 as a better candidate, adopted by 2, and deleted by 3.

Figure 3.3 shows the corresponding concurrent run. Its elements are ordered

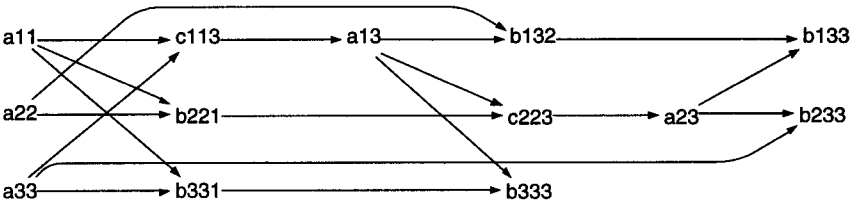


Figure 3.3.

from left to right, with left the earlier and right the later transition occurrences. The upper, middle and lower horizontal lines show the "lifeline" of the agents $1, 2$ and 3, respectively. The remaining arcs denote causal precedence due to messages.

Different interleaved runs may correspond to the same concurrent run; each total extension of the partial order of a concurrent run is an interleaved run, and every interleaved run can this way be obtained from a concurrent run.

As a further step, to ease construction of concurrent runs and to support formal reasoning, it is worthwhile to include the corresponding local states in between each neighboring transition occurrences, as well as before the minimal and behind the maximal elements. Figure 3.4 shows the respective extension of the concurrent run in Fig. 3.3. For a place p, puv denotes a local state, with token (u, v) at p. Shorthands for places, as introduced in Fig. 3.4, will be applied in the rest of this chapter.

Unordered local states may arise together in an interleaved run. Even more, each maximal set of pairwise unordered local states constitutes a global state of the corresponding interleaved runs.

3.4 Progress on concurrent runs

Here we consider liveness properties that are based on concurrent runs. In analogy to the leads-to operator of Sect. 2.3, a formula

$$p \hookrightarrow q \tag{36}$$

(p *causes* q) states for each concurrent run K: If p holds at a global state s_0 of K then there exists a state s_1, reachable from s_0 in K, where q holds. Stated differently, (36) holds in a concurrent run K iff there exists at least one interleaving of K (as defined in Sect. 3.3), at which $p \mapsto q$ holds.

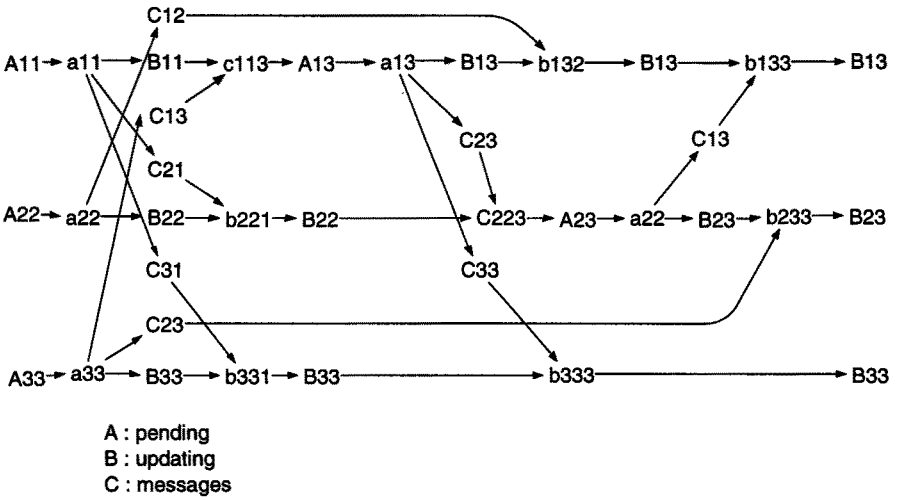


Figure 3.4.

To complete the definition, $p \hookrightarrow q$ is said to *hold in a system* Σ iff $p \hookrightarrow q$ holds for each concurrent run of Σ.

As a first example, assume any instantiation of U and W in Fig. 3.1 (which then fixes V and M). Then a token (u, w) at *pending* enables the transition a in mode $x = u$ and $y = w$. There exists no other transition that could engage the token; hence $a(u, w)$ will occur. With shorthands of Fig. 3.4 this is written

$$A.(u, w) \overset{a(u,v)}{\hookrightarrow} B_1.u. \tag{37}$$

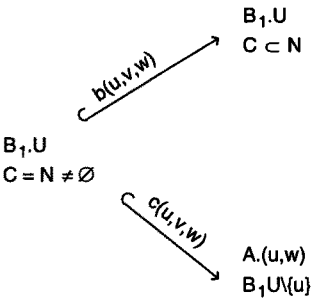
Replacing the causes operator $\hookrightarrow$ by the leads-to operator $\mapsto$ would (37) render valid in $\Sigma_{3.1}$. But (37) can be embedded into a *context*. Assume a global state s in a concurrent run K where $A.(u, w) \wedge B_1.U \setminus \{u\}$ holds. Again, s enables $a(u, w)$, among many other transitions, but in K we consider occurrence of $a(u, w)$, obtaining

$$\begin{array}{l} A.(u, w) \\ \wedge B_1.U \setminus \{u\} \end{array} \overset{a(u,w)}{\hookrightarrow} B_1.U. \tag{38}$$

$B_1.U \setminus \{u\}$ is the *context* of (38). Replacing $\hookrightarrow$ by $\mapsto$ in (38) in general invalidates the formula in $\Sigma_{3.1}$, as there may be an interleaving with a transition that invalidates $B_1.U \setminus \{u\}$ before $a(u, w)$ would occur.

As a second example, let s be a global state such that $B_1.U \wedge C = N \neq \emptyset$ holds at s, with shorthands B and C as in Fig. 3.4 and some set $N \subseteq M(U)$ of messages. $N \neq \emptyset$ implies some $u, w_0 \in U$ with $C.(u, w_0)$, and $B_1.U$ implies some $v \in U$ with $B.(u, v)$. With the valuation $x = u$, $y = v$ and $z = w$, the state s enables $b(u, v, w_o)$ or $c(u, v, w_0)$. There may be some other message (u, w)

in C which u engages in (instead of engaging in w_0). But it is guaranteed that $b(u,v,w)$ or $c(u,v,w)$ will occur, for some $w \in U$. This is graphically represented as



(39)

3.5 Fundamental state properties

An obvious place invariant implies that each site is either *pending* or *updating*:

$$A_1 + B_1 = U. \tag{40}$$

Furthermore, a site v, already knowing the leader, is related to its neighbors by a property derived from a trap. To this end, assume a state s and two neighboring sites $u, v \in U$, and $s \models B.(u, \max)$. s has been reached by occurrence of $a(u, \max)$. This action also produced $C.(v, \max)$. With s considered as (a new) initial state, an initialized trap yields the inequality $A.(u, \max) + C.(v, \max) + A.(v, \max) + B.(v, \max) \geq 1$. Together with (40) this yields the valid propositional formula

$$B.(u, \max) \vee C.(v, \max) \vee A.(v, \max) \vee B.(v, \max). \tag{41}$$

Intuitively formulated, each neighbor of a site already updating with the leader is also aware of the leader, or a corresponding message is pending.

3.6 A fundamental progress property

A weight function f will be required, that assigns each state (u, v) its "better" candidates. So, for all $u, v \in U$ let

$$f(u,v) = \{(u,w) \mid w \in U \wedge w > v\}. \tag{42}$$

Obviously, $f(u,v) = \emptyset$ if $v = \max$.

Now, let us consider a state in which all sites are updating (i.e. $B_1.U$), $f(B) = M$ for some $M \neq \emptyset$ and $C = N$ for some set of messages N. In such a state some site not yet knowing the leader will eventually find a better candidate or will consume one of its pending messages. Thus, eventually a state in which all sites are updating and $(C \subset N_1 \wedge f(B) = M) \vee f(B) \subset M$ will be reached. This is verified by the proof graph of Fig. 3.5. The proof graph's nodes are justified as follows:

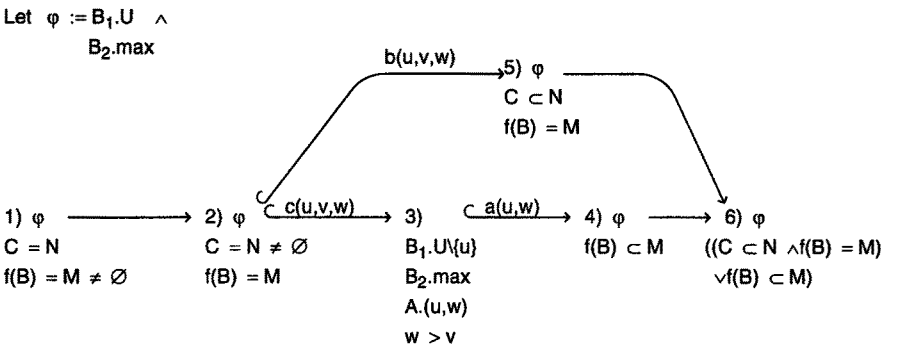


Figure 3.5. Proof graph for $\Sigma_{3.1}$

node 1: $B_2.\max$, $f(B) \neq \emptyset$ and the graph's connectedness imply neighboring sites u and w, $B.(u, \max)$, and $B.(v, i)$ with $i < \max$. Then $C.(u, \max)$ by (41) and (40).

node 2: $C \neq \emptyset$ implies some $C.(u, w)$, and φ implies some $B.(u, v)$. This enables the occurrence of $b(u, v, w)$ or $c(u, v, w)$.

node 3: by the occurrence rule.

nodes 4 and 5: propositional logic.

3.7 Proof of (34)

The proof graph in Fig. 3.5 shows

$$\begin{array}{ll} \varphi & \hookrightarrow \quad \varphi \\ C = N & \quad\quad ((C \subset N \wedge f(B) = M)\,. \\ f(B) = M \neq \emptyset & \quad\quad \vee f(B) \subset M) \end{array} \tag{43}$$

C may shrink finitely often only, hence finitely many iterations of (43) yield

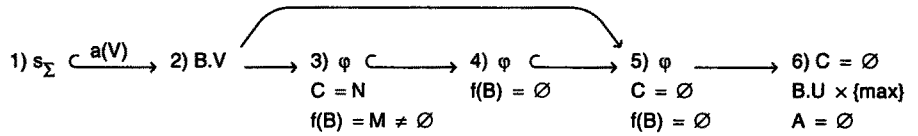
$$\begin{array}{ll} \varphi & \hookrightarrow \quad \varphi \\ C = N & \quad\quad f(B) \subset M\,. \\ f(B) = M \neq \emptyset & \end{array} \tag{44}$$

A remaining message is cleared by

$$\begin{array}{ll} \varphi & \hookrightarrow \quad \varphi \\ C = N & \quad\quad C \subset N \quad , \\ f(B) = \emptyset & \quad\quad f(B) = \emptyset \end{array} \tag{45}$$

as $C.(u, v) \wedge f(B) = \emptyset$ implies $C.(u, v) \wedge B.(u, \max)$, hence enables $b(u, v, \max)$.

The following proof graph now proves (34):

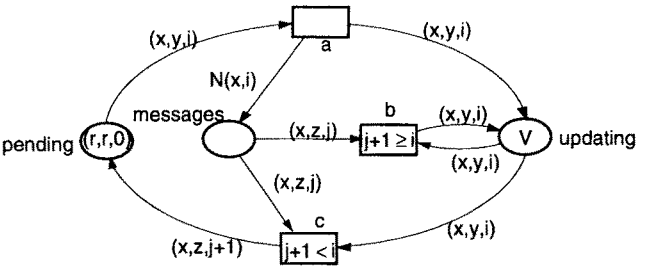


Its nodes are justified as follows:

node 1: by the occurrence of $a(v, v)$ for each $v \in V$.
node 2: propositional reasoning.
node 3: finitely many iterations of (44).
node 4: finitely many iterations of (45).
node 5: by definition of φ and f.

3.8 A variant of the algorithm

The above algorithm terminates with each site holding the leader's name. As a variant, each site will now be informed about its distance to the leader and about a distinguished neighbor closer to the leader. A site then may effectively communicate with the leader along its distinguished neighbor. The respective paths to distinguished neighbors form a *minimal spanning tree* in the underlying network. Figure 3.6 gives the algorithm.



sort	site	var	i, j : nat
sort	state = site × site × (nat ∪ {ω})	var	x, y, z : site
const	⊥, r : site		x,y ∈ U → x W^* y
const	U : set of sites		$W_1 \cup W_2 = U$
const	V : set of states		r ∈ U
const	W : set of (sites × sites)		⊥ ∉ U
	≤ : total order on U		V = {(u,⊥,w) | u ∈ U \ {r}}
fct	N : site × nat → set of states		N(x,y,i) = W(x) × {y} × {i}

Figure 3.6. Shortest distance to a root

Initially, the leader r is pending with itself as a path to the leader candidate, and distance 0 to the leader. All other sites are initially updating with the

unspecified leader candidate $\perp$ and infinite distance. In later phases, a *pending* token (u, v, n) indicates that there is a path of length n from u along v to the leader. A *pending* site u forwards its actual distance n to all its neighbors (by action $a(u, v, n)$) and then turns *updating*. An *updating* token (u, v, n) may receive a message (u, w, m). In case the reported distance m of w to the leader would not improve the actual distance n, the site u remains with distance n along neighbor v (action $b(u, v, w, n, m)$, with ordered set (x, y, z, i, j) of variables). Otherwise u goes pending with distance $m + 1$ along neighbor w (action $c(u, v, w, n, m)$, with ordered set (x, y, z, i, j) of variables).

This algorithm can be generalized to a set $R \subseteq U$ of leaders in the obvious way: Initially, *pending* carries $\{(r, r, 0) \mid r \in R\}$ and *updating* $\{(u, \perp, \omega) \mid u \in U \setminus R\}$. The algorithm then terminates with *updating* triples (u, v, n), where n is the minimal distance to a leader and v the name of a neighbor closer to a leader.

4 Load Balance on Rings

A *service site* is intended to execute *tasks*, provided by the site's environment. At any reachable state a service site has its actual *workload*, i.e., a set of tasks still to be executed. The workload increases or decreases due to interaction with the environment.

Now assume a set of service sites, each one autonomously interacting with its environment. Their individual workload may be heavy or low in a given state, and it is worthwhile to *balance* them: A site with heavy workload may send some tasks to sites with less heavy workload. The overall workload in a set of service sites is *balanced* whenever the cardinality of the workload of two sites differs at most by one.

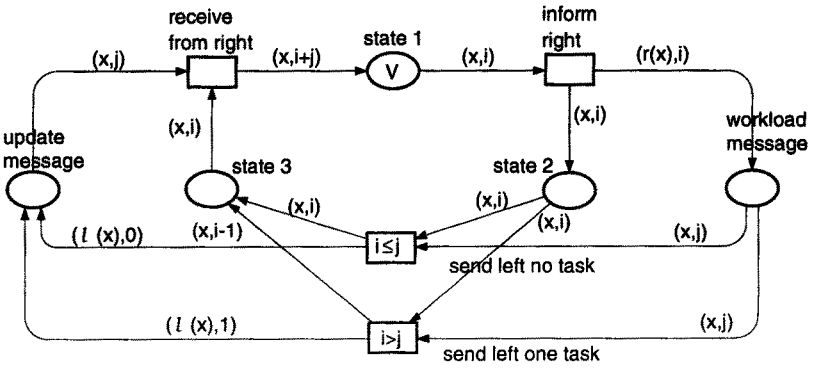
A distributed algorithm is constructed in the following, organizing load balancing in a set of service sites. The communication lines among sites are assumed to form a ring. Each agent u alternately sends a *workload message* to its right neighbor, $r(u)$, and a *task message* to its left neighbor, $l(u)$. A *workload message* of u informs $r(u)$ about the cardinality of the actual workload of u. A *task message* of u depends on the previous workload message of $l(u)$: If this message reports less tasks than u has, the next task message of u transfers one of u's tasks to $l(u)$. Otherwise, the next task message of u transfers no task to $l(u)$. Intuitively formulated, a site u forwards a task to $l(u)$ whenever the workload of u exceeds the workload of $l(u)$.

4.1 A distributed load balance algorithm

Figure 4.1 shows a load balance algorithm with fixed workload: The overall number of tasks remains constant. Each state of a site u is represented as a pair (u, n), with n the cardinality of u's actual workload. The task transfered from u to $l(u)$ by a task message $(l(u), 1)$, is not represented itself.

With the ordered set (x, i, j) of variables, action *inform right* describes communication with right neighbors: A site u with n tasks with action

inform right(u, n, m) receives a task message (u, m) (with $m = 0$ or $m = 1$) from $r(u)$, updates its actual workload, n, and returns a corresponding workload message $(r(u), n + m)$ back to $r(u)$, indicating that u has now $n + m$ tasks. With the same ordered set of variables, actions *send left no task* and *send left*



sort site
sort alloc = site× nat

const U : set of sites
const V : set of alloc
fct l , r : site → site

var i, j : nat
var x,y : site
∀ x ∈ U ∃ i∈ nat : (x,i)∈ V
∀x ∀y ∃ n ∈ nat: y = r^n(x)
l (r(x)) = x

Figure 4.1. Distributed load balance

one task describe communication with left neighbors: A site u with n tasks receives a workload message (u, m) from $l(u)$, compares n and m, and returns a task with action *send left one task*(u, n, m) in case its actual workload, n, exceeds $l(u)$'s reported workload, m. Technically, this is denoted by the inscription $i > j$, which denotes that transition *send left one task* may only occur in a mode where the value assigned to i exceeds the value assigned to j. Otherwise, u sends a task message with *send left no task*(u, n, m), to $l(u)$, containing no task.

Initially, each site u informs $r(u)$ about its actual workload.

4.2 Decisive properties of the algorithm

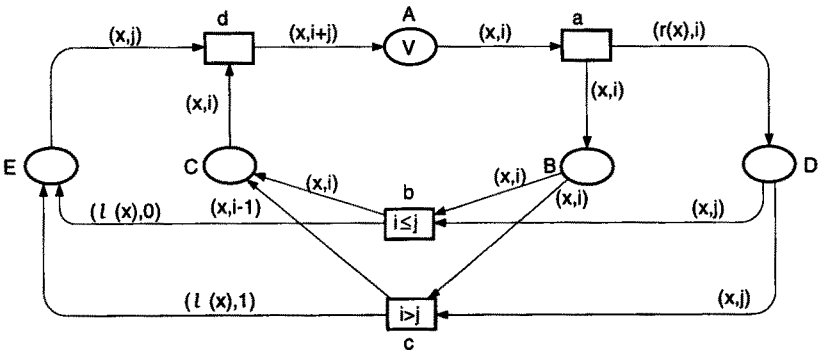
The above algorithm never terminates; each run is infinite. The overall workload is eventually balanced, as described above. Two cases may be distinguished, depending on the overall workload $w := \Sigma_{v \in V_2} v$ and the number $|U|$ of sites:

In case w is a multiple of $|U|$, a state will be reached where transition *send left one task* remains inactive forever, and *state1*, *state2*, and *state3* together contain the tokens (u, n) with $u \in U$ and $n = \frac{w}{|U|}$. Otherwise a state will be reached where

for all tokens (u, n) and (v, m) in *state1*, *state2*, and *state3* holds $|m-n| \leq 1$, and this remains valid forever. The algorithm behaves quite regularly: With initially V at *state1*, it evolves exactly one concurrent run. This run is strictly organized in rounds: All sites concurrently execute action *inform right* and produce a workload message for their respective right neighbor. Then all sites concurrently execute *send left no task* or *send left one task*, thus producing a task message for their respective left neighbor. Finally, *receive from right* completes a round.

4.3 Properties to be proven

Figure 4.2 is a redrawn version of the distributed load balance algorithm. We have to show that the overall workload remains constant, eventually is balanced, and henceforth remains balanced.



sort site
sort alloc = site× nat

const U : set of sites
const V : set of alloc
fct l , r : site → site

var i, j : nat
var x,y : site
∀ x ∈ U ∃ $_1$i∈ nat : (x,i)∈ V
∀x ∀y ∃ n ∈ nat: y = r^n(x)
l (r(x)) = x

Figure 4.2. Renamed distributed load balance

A formal representation of those properties in terms of $\Sigma_{4.2}$ can be based on the following functions. For any place $p \in \{A, B, C, E\}$ and any site $u \in U$, let

$$\begin{aligned} \sigma(p,u) &:= \begin{cases} 0 & \text{iff } \neg p_1.u \\ n & \text{iff } p.(u,n) \end{cases}, \\ \sigma(u) &:= \Sigma_{p\in\{A,B,C,E\}}\, \sigma(p,u), \text{ and} \\ \sigma &:= \Sigma_{u\in U}\, \sigma(u). \end{aligned} \tag{46}$$

These functions describe the workload of site u at place p, the entire workload of u and the overall workload in the system, respectively. The initial overall workload is k iff $a_{\Sigma_{4.2}} \models \sigma = k$. A *balanced* state meets the predicate

$$\text{balanced} := u, v \in U \to |\sigma(u) - \sigma(v)| \leq 1. \tag{47}$$

So we have to show the state property

$$\Sigma_{4.2} \models \sigma = k \tag{48}$$

and the progress property

$$\Sigma_{4.2} \models a_\Sigma \mapsto \text{balanced}. \tag{49}$$

Furthermore, we have to show that all states reachable from a balanced state are balanced, i.e., for each step $r \xrightarrow{t} s$,

$$\text{balanced}(r) \to \text{balanced}(s). \tag{50}$$

4.4 Place invariants

We have two quite obvious place invariants. First, each site is always in one of the three *states* of $\Sigma_{4.1}$ (together with its token load): $A_1 + B_1 + C_1 = U$ (with $U = V_1$ according to the specification of Fig. 4.2). Hence in particular for each $u \in U$ holds

$$A_1.l(u) + B_1.l(u) + C_1.l(u) = 1. \tag{51}$$

Second, each site is either in the quiet *state1* or has sent a workload message to its right neighbor (i.e., is the left neighbor of the first component of a workload message), or is to receive an update message: $A_1 + r(D_1) + E_1 = U$. Hence for each $u \in U$ follows $A_1.l(u) + r(D_1).l(u) + E_1.l(u) = 1$, which in turn yields

$$A_1.l(u) + D_1.u + E_1.l(u) = 1. \tag{52}$$

4.5 Further properties of the algorithm

Here we observe and exploit a particular kind of *regular* behavior of $\Sigma_{4.1}$, that in a stronger version has been the essence of the phase synchronization algorithm, $\Sigma_{2.2}$. In each sequential run of $\Sigma_{2.2}$, each reachable state is eventually followed by a state where all sites are *busy*. A similar, weaker property holds in $\Sigma_{4.1}$: In each *concurrent* run, each reachable state is eventually followed by a state where all sites are in their local state *state 1*. This means that in $\Sigma_{4.2}$ holds the formula true $\hookrightarrow$ "$AA_1.U$. Generally, a formula p is a *ground formula* of a system net Σ iff true $\hookrightarrow p$ holds at Σ.

Two basic properties are required in the following: the ground formula $A_1.U$, and an upper bound for the workload of the sender of a workload message. To start with, we first show:

$$A_1.U \text{ is a ground formula.} \tag{53}$$

Upon proving (53), observe that all steps starting at $A_1.U$ are shaped $A_1 \xrightarrow{a(u,n)} A_1.U - u \wedge B_1.u \wedge D_1.r(u)$, for some $u \in U$ and $n \in \mathbb{N}$. Then (53) follows from Theorem 18.2 and the following proof graph:

1) $A_1.U - u \wedge B_1.u \wedge D_1.r(u) \hookrightarrow$
2) $B_1.U \wedge D_1.U \hookrightarrow$
3) $C_1.U \wedge E_1.U \hookrightarrow$
4) $A_1.U$.

Its nodes are justified as follows:

1) by occurrence of $a(v,n)$ for all $(v,n) \in V, v \neq u$
2) by occurrence of $b(v,n,m)$ or $c(v,n,m)$ for all $v \in U$
3) by occurrence of $d(v,n,m)$ for all $v \in U$.

Second, we show that a workload message tops its sender's token load:

$$D.(u,n) \rightarrow \sigma(l(u)) \leq n. \tag{54}$$

(54) is obviously true at the initial state because $D = \emptyset$. Inductively assume a step $r \xrightarrow{t} s$ with $r \models (54)$. Upon proving $s \models (54)$ two cases are distinguished:

i. Assume $r \not\models D_1.u$ and $s \models D.(u,n)$. Then $t = a(l(u),n)$ (by the structure of the net). Then $s \models B.(l(u),n)$ (by the occurrence rule). Hence $s \models B_1.l(u)$, hence $s \models \neg A_1.l(u) \wedge \neg C_1.l(u)$, by (51). Furthermore, the assumption of $s \models D.(u,n)$ implies $s \models D_1.u$, hence $s \models \neg E_1.l(u)$ (by (52)). Both arguments together imply $\sigma(l(u)) \leq \sigma(B.l(u))$. Then $s \models B.(l(u),n)$ implies the proposition.
ii. Assume $r \models \sigma(l(u)) \leq n$ and $s \not\models \sigma(l(u)) \leq n$. Then $t = c(u,n,m)$, for some $n, m \in \mathbb{N}$ (by the structure of the net). Then $s \models E_1.(l(u),n)$ (by the occurrence rule). Then $s \models \neg D_1.(u,n)$ (by (52)), hence the proposition.

4.6 A decreasing weight

A weight function τ on states will be employed, defined for each state s of $\Sigma_{4.2}$ by $\tau(s) = n$ iff $s \models$

$$\Sigma_{u \in U} \sigma(u)^2 = n. \tag{55}$$

It will turn out that no step increases τ. Furthermore, τ decreases upon occurrence of $c(u,n,m)$, provided $m+1$ is smaller than n.

First we show that c does not increase τ: Let $r \xrightarrow{c(u,n,m)} s$ be a step. Then

$$\tau(r) \geq \tau(s). \tag{56}$$

In order to show (56), observe that at r holds $^{(*)}$ $B.(u,n)$ as well as $^{(**)}$ $D.(u,m)$, due to the occurrence rule. Furthermore, with $r \models \sigma(l(u)) = a \wedge \sigma(u) = b$, at r

holds $b \geq n$ by $^{(*)}$, $n > m$ by inscription of transition c, and $m \geq \sigma(l(u))$, by $^{(**)}$ and (54); hence $^{(***)}$ $(a - b + 1) \leq 0$. Now,

$$\begin{aligned}\tau(s) &= \tau(r) - a^2 - b^2 + (a+1)^2 + (b-1)^2 \text{ (by the structure of } c(u,n,m)) \\ &= \tau(r) - a^2 - b^2 + a^2 + 2a + 1 + b^2 - 2b + 1 \\ &= \tau(r) + 2(a - b + 1) \\ &\leq \tau(r), \text{ by } ^{(***)}, \text{ hence (56).}\end{aligned}$$

(56) can be strengthened in case $\sigma(u) > \sigma(l(u)) + 1$: Let $r \xrightarrow{c(u,n,m)} s$ be a step of $\Sigma_{4.2}$ with $\sigma(u) > \sigma(l(u)) + 1$. Then

$$\tau(r) > \tau(s). \tag{57}$$

Proof of (57) is a slight variant of the above proof graph of (56): $b > a + 1$ now implies $(a - b + 1) < 0$. Then the last two lines read $\tau(r) + 2(a - b + 1) < \tau(r)$. Generalizing (56), no step at all increases τ: Let $r \xrightarrow{t} s$ be a step of $\Sigma_{4.2}$. Then

$$\tau(r) \geq \tau(s). \tag{58}$$

To prove (58), observe that $\tau(r) \neq \tau(s)$ implies $t = c(u,n,m)$ for some $u \in U$ and $n, m \in \mathbb{N}$, by definition of τ and σ, and the structure of $\Sigma_{4.2}$. Then (58) follows from (56).

4.7 Descents

A descent of length k consists of a sequence $u, l(u), l^2(u), \ldots, l^{k+1}(u)$ of sites, with token loads decreasing by 1 from u to $l(u)$ and by any number from $l^k(u)$ to $l^{k+1}(u)$, and identical token load of $l(u), \ldots, l^k(u)$. More precisely, for any site $u \in U$ and any state s, the *descent of u at s* amounts to k (written: $\delta(u) = k$) iff there exists some $n \in \mathbb{N}$ with

$$\begin{aligned}&\sigma(u) = n + 1, \\ &\sigma(l^i(u)) = n \quad (i = 1, \ldots, k), \\ &\sigma(l^{n+1}(u)) \leq n - 1.\end{aligned} \tag{59}$$

Figure 4.3 outlines examples.

In general, there may exist states s with undefined descent $\delta(u)$. Even more, obviously holds for all states s of $\Sigma_{4.2}$:

$$s \text{ is balanced iff no site has a descent at } s. \tag{60}$$

In the sequel we will show that large descents reduce to small ones and small descents reduce the weight τ. Each large descent reduces to a smaller one, as exemplified in Fig. 4.3.

$$A_1.U \wedge \delta(u) = k \wedge k \geq 2 \hookrightarrow A_1.U \wedge \delta(l(u)) = k - 2. \tag{61}$$

This proposition follows from the following proof graph:

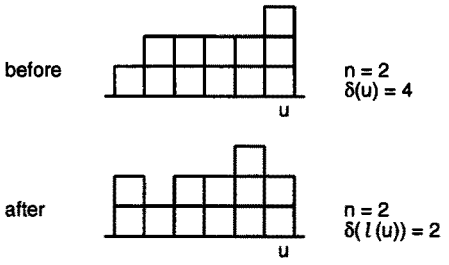


Figure 4.3. Reduction of a large descent

1) $A_1.U \wedge \delta(u) = k \wedge k \geq 2 \rightarrow$
2) $A_1.U \wedge A.(u, n+1) \wedge A.(l^i(u), n)\ (i = 1, \ldots, k)\ \wedge A.(l^{k+1}(u), n-j) \hookrightarrow$
3) $B_1.U \wedge D_1.U \wedge B.(u, n+1) \wedge B.(l^i(u), n)\ (i = 1, \ldots, k)\ \wedge D.(u, n) \wedge D.(l^i(u), n)\ (i = 1, \ldots, k-1)\ \wedge D.(l^k(u), n-j) \hookrightarrow$
4) $C_1.U \wedge E_1.U \wedge C.(l^i(u), n)\ (i = 1, \ldots, k-1) \wedge C.(l^k(u), n-j) \wedge E.(l(u), 1) \wedge E.(l^i(u), 0)\ (i = 2, \ldots, k)\ \hookrightarrow$
5) $A_1.U \wedge A.(l(u), n+1) \wedge A.(l^i(u), n)\ (i = 2, \ldots, k-1)\ \wedge A.(l^k(u), n-1) \rightarrow$
6) $A_1.U \wedge \delta(l(u)) = k - 2.$

Its nodes are justified as follows:

node 1: there exist $n, j \geq 1$ with the described properties, according to (59)
node 2: by occurrence of $\{a(v, m) \mid v \in U \wedge A.(v, m)\}$
node 3: by occurrence of $c(u, n+1, n)$, $b(l^i(u), n, n)$ for $i = 1, \ldots, k-1$, $c(l^k(u), n, n-j)$, and $b(v, m, m')$ or $c(v, m, m')$ for all $v \neq l^i(u)$ $(i = 0, \ldots, k)$
node 4: by occurrence of $\{d(v, m, m') \mid v \in U \wedge C.(v, m) \wedge E.(v, m')\}$
node 5: by (59).

Each descent of length 0 reduces the weight τ, as outlined in Fig. 4.4.

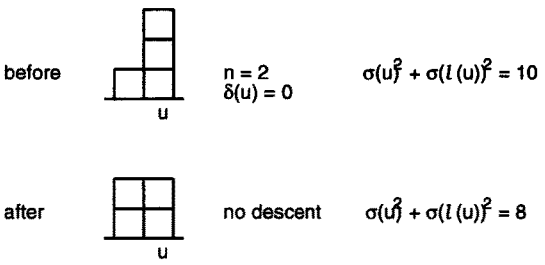


Figure 4.4. Descent of length 0

Formally,

$$A_1.U \wedge \delta(u) = 0 \wedge \tau = m \hookrightarrow \tau < m. \tag{62}$$

This proposition follows from the following proof graph:

1) $A_1.U \wedge \delta(u) = 0 \wedge \tau = m \rightarrow$
2) $A.(u, n+1) \wedge A.(l(u), n-j) \wedge \tau = m \hookrightarrow$
3) $B.(u, n+1) \wedge D.(u, n-j) \wedge \tau \leq m \overset{c(u,n+1,n-j)}{\hookrightarrow}$
4) $\tau < m.$

Its nodes are justified as follows:

node 1: there exist $n, j \geq 1$ with the described properties, according to (59)
node 2: by occurrence of $a(u, n+1)$ and $a(l(u), n-j)$
node 3: by (56).

Each descent of length 1 likewise reduces the weight τ, as outlined in Fig. 4.5.

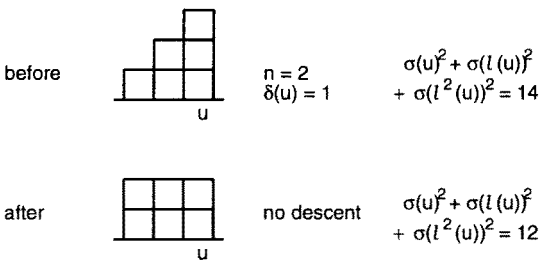


Figure 4.5. Descent of length 1

Formally,

$$A_1.U \wedge \delta(u) = 1 \wedge \tau = m \hookrightarrow \tau < m. \tag{63}$$

This proposition follows from the following proof graph:

1) $A_1.U \wedge \delta(u) = 1 \wedge \tau = m \rightarrow$
2) $A_1.U \wedge A.(u, n+1) \wedge A.(l(u), n) \wedge A.(l^2(u), n-j) \wedge \tau = m \hookrightarrow$
3) $B.(u, n+1) \wedge D.(u, n) \wedge B.(l(u), n) \wedge D.(l(u), n-j) \wedge \tau \leq m$
4) $B.(u, n+1) \wedge D.(u, n) \wedge B.(l(u), n) \wedge D.(l(u), n-j) \wedge \tau \leq m \wedge \sigma(u) = \sigma(l(u)) + \wedge \overset{c(l(u),n,n-j)}{\hookrightarrow}$
5) $B.(u, n+1) \wedge D.(u, n) \wedge \tau \leq m \wedge \sigma(u) = \sigma(l(u)) + 2 \overset{c(u,n+1,n)}{\hookrightarrow}$
6) $\tau < m.$

Its nodes are justified as follows:

node 1: there exist $n, j \geq 1$ with the described properties, according to (59)
node 2: by occurrence of $a(u, n+1)$, $a(l(u), n)$, and $a(l^2(u), n-1)$
node 3: by (51), and (52)
node 4: by the occurrence rule
node 5: by the occurrence rule, and (57).

The weight τ is reducible as long as there exists a descent:

$$\tau = m \hookrightarrow \tau < m \vee \forall u \in U : \delta(u) \text{ is undefined.} \tag{64}$$

. This proposition follows from the following proof graph:

1) $\tau = m \hookrightarrow$
2) $A_1.U \wedge \tau \leq m \rightarrow$ 3) $\forall u \in U : \delta(u)$ undefined
4) $A_1.U \wedge \tau \leq m \wedge \exists u \in U, k \in \mathbb{N}$ with $\delta(u) = k \hookrightarrow$
5) $A_1.U \wedge \tau \leq m \wedge \exists u \in U$ with $\delta(u) \leq 1 \hookrightarrow$
6) $\tau < m \rightarrow$
7) $\tau < m \vee \forall u \in U : \delta(u)$ undefined

Its nodes are justified as follows:

node 1: by (53) and (58)
node 2: propositional logic
node 3: propositional logic
node 4: by $\lfloor \frac{k}{2} \rfloor$ fold application of (61)
node 5: by (62) if $\delta(u) = 0$, and by (63) if $\delta(u) = 1$
node 6: propositional logic.

4.8 Proof of the essential properties

To show (48), let $r \xrightarrow{t} s$ be any step of $\Sigma_{4.2}$, and assume $\sigma_r = k$. Then $\sigma_s = k$ follows due to the structure of $\Sigma_{4.2}$. Finally, (48) follows by induction on the length of interleaved runs of $\Sigma_{4.2}$.

To prove (50), first consider the case of $t = c(u, n, m)$ for some $u \in U$ and $n, m \in \mathbb{N}$. Then at r holds $B.(u, n) \wedge D.(u, m) \wedge n > m$. Furthermore, $\sigma(u) \geq n$ by (46) and $m \geq \sigma(l(u))$, by (54). Hence $\sigma(u) = n$ and $\sigma(l(u)) = n - 1$, as r is balanced. Then at s holds $\sigma(u) = n - 1$ and $\sigma(l(u)) = n$. The workload $\sigma(v)$ remains unchanged for all $v \neq u$. Hence s is balanced, too.

All actions t not involving c do not touch $\sigma(u)$ for any $u \in U$, hence the proposition.

Proof of (49) requires

$$a_\Sigma \hookrightarrow \text{balanced} \tag{65}$$

proven by the following proof graph:

$$a_\Sigma \to \tau = m \hookrightarrow \tau = n_1 < m \hookrightarrow \tau = n_2 < n_1 \hookrightarrow \cdots \hookrightarrow \tau = n_m = 0$$

$$\forall u \in U : \delta(u) \text{ is undefined}$$

$$\downarrow$$

balanced

which is justified as follows: The first implication states that τ has some value, m, at the initial state a_Σ. All other nodes in the upper line are justified by (64). The last implication holds by (60).

In order to show (49), let w be an interleaved run of $\Sigma_{4.2}$. Then there exists a concurrent run K of $\Sigma_{4.2}$, including all actions of w. K has a reachable, balanced state, s, (by (65)). Then w has a reachable state, s', such that all actions of K, occurring before s, are actions of w, occurring before s'. Then $\Sigma_{4.2} \models s \mapsto s'$ and s' is balanced by (50), hence the proposition.

5 The Echo Algorithm

5.1 The problem

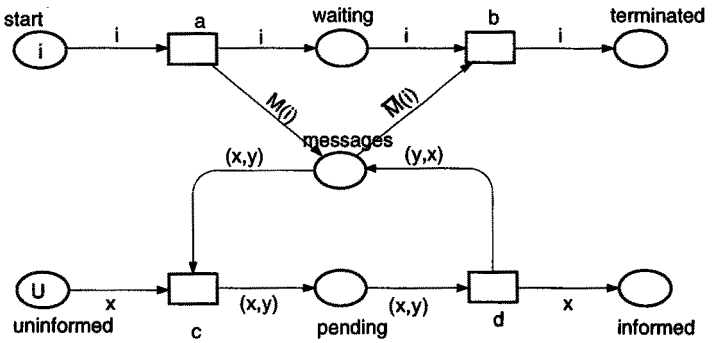
Given a finite, connected network with a particular initiator site, the echo algorithm organizes acknowledged broadcast of the initiator's message throughout the entire network to all sites: The initiator will terminate only after all other sites are informed.

Figure 5.1 shows one round of messages, sent by the initiator i to all its neighbors. Messages and receipts are jointly represented in one place. The central idea of the echo algorithm is now covered in the step from $\Sigma_{5.1}$ to $\Sigma_{5.2}$: Upon receiving the initiator's message, a neighbor of the initiator forwards the message to all its neighbors except for the initiator, and remains pending until receiving messages from all those neighbors. Each site is eventually addressed in this schema. Each uninformed site $u \in U$ receives in general more than one message, hence u selects one occurrence mode (u, v) of action c. In this case, v is called the *parent site* of u. The pairs (u, v) with v the parent site of u, form *a spanning tree* in the underlying network: For each site $u \in U$ there exists a unique sequence $u_0 \ldots u_n$ of sites with $u_0 = u$, $u_n = i$ and u_i the parent site of u_{i-1} $(i = 1, \ldots, n)$. A site u is a *leaf* of the spanning tree if no neighbor of u elects u as its parent node.

For each pending leaf (u, v), the place *messages* eventually holds all messages $\overline{M}(u) - (u, v)$, hence the leaf becomes *informed* by occurrence of d in mode (u, v). The leaves are the first to become (concurrently) *informed.* Then all sites are consecutively *informed*, causally ordered along the spanning tree. Finally, the initiator's transition b is enabled, and the *waiting* initiator turns *terminated.*

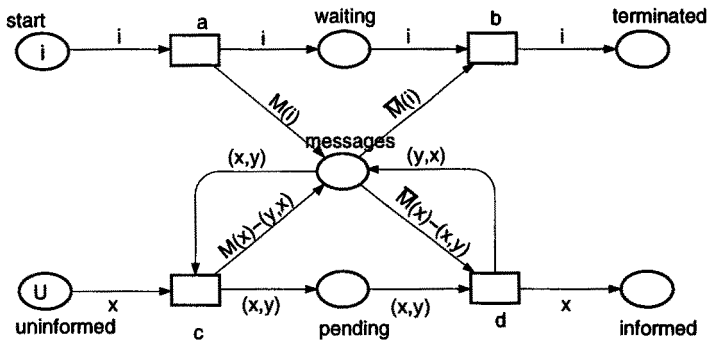
5.2 Properties to be proven

Figure 5.3 provides a redrawn version of the Echo Algorithm of Fig. 5.2. It has



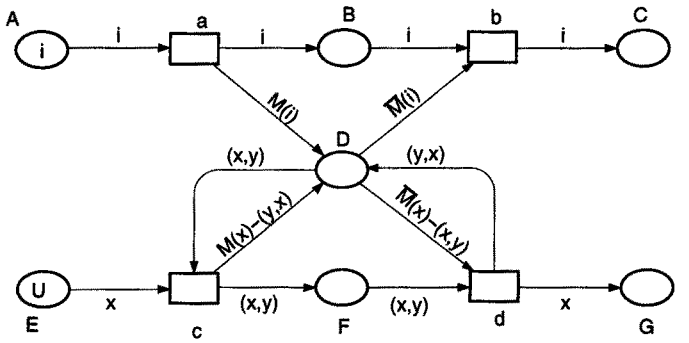
sort	site	$W = W^{-1}$
sort	message = site × site	$x,y \in U \cup \{i\} \rightarrow x\, W^* y$
const	i : site	$W_1 = U \cup \{i\}$
const	U : set of sites	$i \notin U$
const	W : set of (sites × sites)	$M(x) = W(x) \times \{x\}$
fct	M, $\overline{M}$: site → set of messages	$\overline{M}(x) = M(x)^{-1}$
var	x,y : site	

Figure 5.1. The initiator informs its neighbors



sort	site	$W = W^{-1}$
sort	message = site × site	$x,y \in U \cup \{i\} \rightarrow x\, W^* y$
const	i : site	$W_1 = U \cup \{i\}$
const	U : set of sites	$i \notin U$
const	W : set of (sites × sites)	$M(x) = W(x) \times \{x\}$
fct	M, $\overline{M}$: site → set of messages	$\overline{M}(x) = M(x)^{-1}$
var	x,y : site	

Figure 5.2. The echo algorithm



sort	site	$W = W^{-1}$
sort	message = site × site	$x,y \in U \cup \{i\} \rightarrow x\,W^*\,y$
const	i : site	$W_1 = U \cup \{i\}$
const	U : set of sites	$i \notin U$
const	W : set of (sites × sites)	$M(x) = W(x) \times \{x\}$
fct	M, $\overline{M}$: site → set of messages	$\overline{M}(x) = M(x)^{-1}$
var	x,y : site	

Figure 5.3. Redrawn echo algorithm $\Sigma_{5.1}$

two decisive properties: Firstly, the initiator terminates only if all other sites have been informed before. In Fig. 5.3, this reads

$$C.i \rightarrow G.U. \tag{66}$$

Secondly, the initiator *will* eventually terminate, i.e.,

$$s_{\Sigma_{5.3}} \mapsto C.i. \tag{67}$$

Both (66) and (67) will be verified in the sequel.

There is no straight-forward place invariant or trap that would prove (66). Nor is there an intuitively convincing proof graph for (67). Rather, one has to argue inductively along a spanning tree that yields at place F.

5.3 Three place invariants

Figure 5.3 has three important place invariants, as given in Fig. 5.4. Two of them are intuitively quite obvious, representing the "life lines" of the initiator i and of all other sites, respectively.

The equation of I_1 is $A + B + C = i$. This implies

$$A.i + B.i + C.i = 1, \tag{68}$$

hence the initiator is always either at its start or is waiting, or is terminated. The equation furthermore implies

	a	b	c	d	s_Σ	I_1	I_2	I_3
A	$-i$				i	A		$M(\mathtt{A})$
B	i	$-i$				B		
C		i				C		$\overline{M}(\mathtt{C})$
D	$M(i)$	$-\overline{M}(i)$	$M(x)$	$-\overline{M}(x)$				D
			$-(x,y)$	$+(x,y)$				
			$-(y,x)$	$+(y,x)$				
E			$-x$		U		E	$M(\mathtt{E})$
F			(x,y)	$-(x,y)$			$\mathtt{F}_1$	$\mathtt{F}+\overline{\mathtt{F}}$
G				x			G	$\overline{M}(\mathtt{G})$
$I \cdot s_\Sigma$						i	U	$M(U')$

Let $\overline{\mathtt{F}} = \mathtt{F}^{-1}$ and $U' = U \cup \{I\}$

Figure 5.4. Matrix, initial state, and three place invariants of $\Sigma_{5.3}$

$$\forall x \in U : A.x + B.x + C.x = 0, \tag{69}$$

hence no non-initiator site ever finds at A, B, or C.

Correspondingly, the equation of I_2 is $E + F_1 + G = U$. This implies

$$\forall x \in U : E.x + F_1.x + G.x = 1, \tag{70}$$

hence each non-initiator is always either uninformed or pending or informed. The equation furthermore implies

$$\forall x \notin U : E.x + F_1.x + G.x = 0, \tag{71}$$

hence the initiator never finds on E, F, or G.

I_3, finally, represents the *potential messages* of the system. Its equation is $M(A) + \overline{M}(C) + D + M(E) + F + \overline{F} + \overline{M}(G) = M(U')$, implying for each message $(y,x) \in M(U')$ the property $M(A).(y,x) + \overline{M}(C).(y,x) + D.(y,x) + M(E).(y,x) + F.(y,x) + \overline{F}.(y,x) + \overline{M}(G).(y,x) = M.(y,x)$, which in turn reduces to

$$\begin{aligned} &\forall x \in U' \quad \forall y \in W(x) : \\ &\quad A.x + C.y + D.(y,x) + E.x + F.(y,x) + F.(x,y) + G.y = 1. \end{aligned} \tag{72}$$

Hence for each message (y,x) holds: Its sender x is still starting or uninformed, or the message has already been sent but not received yet, or one of y and x has received the message from x to y, respectively, or the message's receiver y is terminated or informed.

5.4 The pending site's rooted tree

A further state property will be required, stating that the tokens on F always form a tree with root i. This will be formulated with the help of the following notation:

A sequence $u_0 \ldots u_n$ of sites $u_i \in U'$ is a *sequence of F at a state* s iff $s \models F.(u_{i-1}, u_i)$ for $i = 1, \ldots, n$. (73)

For each reachable state s we will now prove the following two properties:

For each $F_1.u$ there is a unique sequence $u_0 \dots u_n$ of F with $u_0 = u$ and $u_n = i$, (74)

and

the elements of each sequence of F are pairwise different. (75)

Both properties now are together shown by induction on the reachability of states:
Both (74) and (75) hold initially, as $s_{\Sigma_{5.3}} \models F = \emptyset$. Now, let r be a reachable state, let $r \xrightarrow{m} s$ be a step of some transition t, and inductively assume (74) and (75) for r.

The case of $t = a$ or $t = b$ implies $r(F) = s(F)$, hence the step $r \xrightarrow{m} s$ retains both (74) and (75) for s. For $t = c$ or $t = d$ let $m(x) = u$ and $m(y) = v$.

The case of $t = c$ goes as follows: Enabledness of $c(m)$ at r now for r implies $D.(u, v)$ and $E.u$. Then $r \models F_1.v$, according to the following sequence of implications:

1.	⟶	2.	⟶	3.	⟶	4.	⟶	5.
$D.(u,v)$		$D.(u,v)$		$\neg E.v$		$\neg E.v$		$F_1.v$
$E.u$		$E.u$		$E.u$		$\neg G.v$		
		$v \in W(u)$		$v \in W(u)$				

Its nodes are justified as follows:

node 1: (71);
node 2: (72) with $x = v$, $y = u$;
node 3: (72) with $x = u$, $y = v$;
node 4: (70).

Now, $r \models F_1.v$ and the inductive assumption of (74) imply a unique sequence $v \dots i$ of F at state r. Then $uv \dots i$ is a sequence of F at state s, because $s(F) = r(F) + (u, v)$. Together with (70), this implies (74) for s. Furthermore, $r \models u \notin F_1$ (by (70)) and $u \neq i$ by (69), hence (75) for s.

Correspondingly, enabledness of $d(m)$ at r now for r implies $D.\overline{M}(u) - (u, v)$ and $F.(u, v)$. Then $r \models F_2.u$ according to the following sequence of implications:

1.	⟶	2.	⟶	3.	⟶	4.	⟶	5.	⟶	6.
$D.\overline{M}(u)$		$D.\overline{M}(u)$		$F \cap (\overline{M}(u)$		$F \cap (M(u)$		$F \cap M(u) = \emptyset$		$\neg F_2.u$
$-(u,v)$		$-(u,v)$		$-(u,v)) = \emptyset$		$-(v,u)) = \emptyset$				
$F.(u,v)$		$\neg F.(v,u)$		$\neg F.(v,u)$		$\neg F.(v,u)$				

Its nodes 1 and 2 are justified by (72), nodes 3, 4, and 5 by properties of M.

With $r \models \neg F_2.u$, for each sequence $u_0 \dots u_n$ of F, $u_1, \dots, u_n \neq u$. This implies (74) for the state s, because $s(F) = r(F) - (u, v)$. (75) is then trivial, because $s(F) \subseteq r(F)$.

5.5 Proof of the state property (66)

(66) is indirectly proven in three steps:

i. Assume $F \neq \emptyset$. Then there exists some $w \in U'$ with $F.(w,i)$, by (74). Then $\neg C.i$ by (72).

ii. For all $u \in U'$ we show

$$E.u \rightarrow \neg C.i \qquad (*)$$

by induction on the distance of u to i: For $u = i$, (*) holds trivially, as $\neg E.i$ by (71). Inductively assume (*), let $v \in W(u)$, and assume $E.v$. Then $u \in W(v)$, hence $\neg G.u$, by (72). Then $F_1.u$ or $E.u$, by (70). The case of $F_1.u$ implies $F \neq \emptyset$, hence $\neg C.i$ by (i). The case of $E.u$ implies $\neg C.i$ by inductive assumption.

iii. $C.i \rightarrow E = F = \emptyset$, by (i) and (ii). Then (66) follows from (70).

5.6 Progress from *uninformed* to *pending*

Here we show that each *uninformed* site $u \in U$ will eventually go *pending*. In terms of $\Sigma_{5.3}$ this reads:

$$\begin{array}{l}\text{Let } U = V \cup W,\ V \neq \emptyset,\ W \neq \emptyset. \text{ Then} \\ E.V \wedge F_1.W \hookrightarrow \bigvee_{v \in V}(E.V - v \wedge F_1.W + v).\end{array} \qquad (76)$$

This property holds due to the following proof graph:

1) $E.V \wedge F_1.W \wedge V \neq \emptyset \wedge W \neq \emptyset \rightarrow$
2) $E.V \wedge F_1.W \wedge$ ex. $v \in V \wedge$ ex. $w \in W \cup \{i\}$ with $D.(v,w) \hookrightarrow$
3) $E.V - v \wedge F_1.W + v$

Its nodes are justified as follows:

node 1: Connectedness of U' implies some neighbors v, w such that $E.v$, and $F_1.w$ or $w = i$. Furthermore,

i. $F_1.w$ implies $w \in U$ by (71), hence $\neg A.w$ by (69). $w = i$ and $W \neq \emptyset$ imply some $F.(u,i)$ by (74), hence $\neg A.i$ by (72).

ii. $E.v$ implies $v \in U$ by (71), then $\neg C.v$ by (69).

iii. $F_1.w$ implies $\neg E.w$ by (70) and $w = i$ implies $\neg E.w$ by (71).

iv. $E.v$ implies $\neg F_1.v$ by (74), hence $\neg F.(v,w)$.

v. Let $u_0 \ldots u_n$ be a sequence of F with $u_0 = w$ and $u_n = i$, according to (74). The case of $n = 1$ implies $u_1 = i \neq v$, hence $\neg F.(w,v)$. Otherwise, $F_1.u_1$. Then $E.v$ implies $u_1 \neq v$ by (72). Hence $\neg F.(w,v)$.

vi. $E.v$ implies $\neg G.v$ by (70).

Now (i),...,(vi), and (72) imply $D.(v,w)$.

node 2: by the occurrence of $c(v,w)$.

5.7 Progress from *pending* to *informed*

Here we show that each pending site will eventually be informed. In terms of $\Sigma_{5.3}$ this reads:

Let $U = V \cup W$ with $V \neq \emptyset$. Then
$F_1.V \wedge G.W \hookrightarrow \bigvee_{v \in V}(F_1.V - v \wedge G.W + v)$. (77)

This property holds due to the following proof graph:

1) $F_1.V \wedge G.W \wedge V \cup W = U \wedge V \neq \emptyset \rightarrow$
2) ex. $v \in V$ ex. $w \in U$:
   $F_1.V \wedge G.W \wedge V \cup W = U \wedge D.(\overline{M}(v) - (v,w)) \hookrightarrow$
3) ex. $v \in V$ ex. $w \in U$ with $F_1.V - w \wedge G.W + v$.

Its nodes are justified as follows:

node 1: Let $u_0 \ldots u_n$ be a maximal sequence of F. This exists due to (74) and (75). In case u_1 is the only neighbor of u_0, $D.(\overline{M}(u_0) - (u_0,u_1)) = D.((u_0,u_1) - (u_0,u_1)) = D.\emptyset$ which holds trivially. Otherwise, let $(u_0,v) \in \overline{M}(u_0) - (u_0,u_1)$. Then the following six properties hold:
  i. (74) implies some $F.(w,i)$, hence $\neg A.i$ by (72), hence $\neg A.v$ in case $i = v$. Otherwise, $v \in U$, hence $\neg A.v$ by (69).
  ii. $u_0 \in U$ by construction, hence $\neg C.u_0$ by (69).
  iii. $E = \emptyset$ by (70) and $V \cup W = U$, hence $\neg E.v$.
  iv. Maximality of $u_0 \ldots u_n$ implies $\neg F.(v,u_0)$.
  v. $F.(u_0,u_1)$ implies $\neg F.(u_0,v)$ as the path from u_0 to i is unique by (74).
  vi. $F_1.u_0$ implies $\neg G.u_0$.
  Now (i),...,(vi), and (72) imply $D.(u_0,v)$. This argument applies to all $(u_0,v) \in \overline{M}(u_0) - (u_0,u_1)$, hence $D.\overline{M}(u_0) - (u_0,u_1)$.
node 2: by the occurrence of $d(v,w)$.

5.8 Proof of the liveness property (67)

(67) is now proven with the help of the proof graph of Fig. 5.5. Its nodes are justified as follows:

node 1: definition of $s_{\Sigma_{5\,3}}$
node 2: by the occurrence rule
node 3: by the occurrence of $c(u,i)$ with $u \in M(i)$
node 4: $|V|$-fold application of (76)
node 5: $|U|$-fold application of (77)
node 6: we distinguish three cases:
  i. $u \in M(i)$ implies $u \neq i$, hence $\neg A.u$ by (69)
  ii. $G.U$ implies $E = F = \emptyset$ by (70) and (71). Hence $\neg E.u$, $\neg F.(i,u)$, and $\neg F.(u,i)$.
  iii. $i \notin U$ implies $\neg G.i$ by (71).

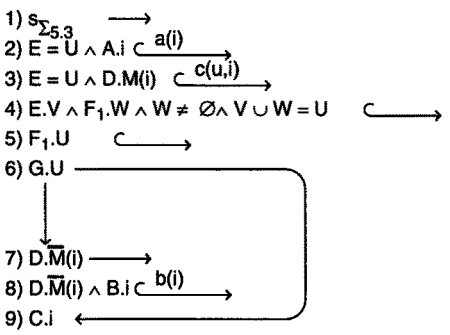


Figure 5.5. A proof graph for $s_{\Sigma_{5.3}} \hookrightarrow C.i$

Now, (i), (ii), and (iii) with (72) imply $D.(i,u) \vee C.i$. This argument applies to all $(i,u) \in \overline{M}(i)$, hence $D.\overline{M}(i) \vee C.i$.

node 7: $\neg C.i$ by (72); $\neg A.i$ because $s_{\Sigma_{5.3}} \rightarrow \neg D.\overline{M}(i)$, the only initial step is $s_{\Sigma_{5.3}} \xrightarrow{a} B.i$, and $\{B.i, C.i\}$ is a trap, initialized after this step. Hence the proposition by (68).

node 8: by the occurrence rule.

Appendix

6 The Concept of System Nets

The conceptual idea of system nets is quite simple: Each place of a system net Σ represents a *set* of local states and each transition of Σ represents a *set* of actions. The sets assigned to the places form the underlying universe:

6.1 Definition *Let Σ be a net. A* universe $\mathcal{A}$ of Σ *fixes for each place $p \in P_\Sigma$ a set $\mathcal{A}_p$, the* domain of p in $\mathcal{A}$.

An *actual state* fixes for each place a subset of its domain. Some algorithmshave reachable states with *multiple* occurrences of elements. Formally, an actual state then fixes a *multiset*. For the sake of simplicity we stick to proper subsets in the following. The canonical generalization to multisets is given in [Weber et al 98]. An *action* correspondingly fixes the degree of change caused by its occurrence:

6.2 Definition *Let Σ be a net with a universe $\mathcal{A}$.*

i. A state *a of Σ assigns to each place $p \in P_\Sigma$ a set $a(p) \subseteq \mathcal{A}_p$.*

ii. Let $t \in T_\Sigma$. An action *m of t assigns to each adjacent arc $f = (p,t)$ or $f = (t,p)$ a set $m(f) \subseteq \mathcal{A}_p$.*

Enabledness and effect of actions, and the notion of steps, are defined as follows:

6.3 Definition *Let Σ be a net with some universe $\mathcal{A}$, let a be a state, let $t \in T_\Sigma$, and let m be an action of t.*

i. m is enabled at a *iff for each place* $p \in {}^\bullet t$, $m(p,t) \subseteq a(p)$ *and for each place* $p \in t^\bullet$, $(m(t,p) \setminus m(p,t)) \subseteq \mathcal{A}_p \setminus a(p)$.

ii. The state $\mathrm{eff}(a,m)$, *defined for each place* $p \in P_\Sigma$ *by*

$$\mathrm{eff}(a,m)(p) := \begin{cases} a(p) \setminus m(p,t) & \text{iff } p \in {}^\bullet t \setminus t^\bullet, \\ a(p) \cup m(t,p) & \text{iff } p \in t^\bullet \setminus {}^\bullet t, \\ (a(p) \setminus m(p,t)) \cup m(t,p) & \text{iff } p \in t^\bullet \cap {}^\bullet t, \\ a(p) & \text{otherwise,} \end{cases}$$

is the effect of the occurrence of m on a.

iii. Assume m is enabled at a. Then the triple $(a, m, \mathrm{eff}(a,m))$ is called a step of t in Σ, *and usually written* $a \xrightarrow{m} \mathrm{eff}(a,m)$.

Steps may be described concisely by means of a canonical extension of actions:

6.4 Proposition *Let Σ be a system net, let $t \in T_\Sigma$, and let $a \xrightarrow{m} b$ be a step of t. Extend m by $m(r,s) := \emptyset$ for all pairs (r,s) of net elements which form no arc of the net. Then for all places $p \in P_\Sigma$, $b(p) = (a(p) \setminus m(p,t)) \cup m(t,p)$.*

A net with a domain for each place and a set of actions for each transition is furthermore equipped with an initial state:

6.5 Definition *A net Σ is a* system net *iff*

i. For each place $p \in P_\Sigma$, a set $\mathcal{A}_p$ is assumed (i.e., a universe of Σ),

ii. for each transition $t \in T_\Sigma$, a set of actions of t *is assumed,*

iii. a state a_Σ is distinguished, called the initial state of Σ.

7 Interleaved and Concurrent Runs

Interleaved runs of system nets can be defined canonically as sequences of steps.

7.1 Definition *Let Σ be a system net and let $a_0 := a_\Sigma$.*

i. For $i = 1, \ldots, n$ assume steps $a_{i-1} \xrightarrow{m_i} a_i$ of Σ such that no action is enabled at a_n. They form a finite interleaved run *w of Σ, written $a_0 \xrightarrow{m_1} a_1 \xrightarrow{m_2} \ldots \xrightarrow{m_n} a_n$. Each $i \in \{0, \ldots, n\}$ is an* index *of w.*

ii. For $i = 1, 2, \ldots$ assume steps $a_{i-1} \xrightarrow{m_i} a_i$ of Σ. They form an infinite interleaved run *w of Σ, sometimes outlined $a_0 \xrightarrow{m_1} a_1 \xrightarrow{m_2} \ldots$. Each $i \in \mathbb{N}$ is an* index *of w.*

Reachable steps, states and actions are defined as follows:

7.2 Definition *Let Σ be a system net.*

i. A step $a \xrightarrow{m} b$ of Σ is reachable in Σ *iff there exists a finite interleaved run $a_\Sigma \xrightarrow{m_1} a_1 \xrightarrow{m_2} a_2 \to \ldots \to a_{n-1} \xrightarrow{m_n} a_n$ with $a_{n-1} \xrightarrow{m_n} a_n = a \xrightarrow{m} b$.*

ii. A state a of Σ is reachable in Σ *iff $a = a_\Sigma$ or there exists a reachable step formed $b \xrightarrow{m} a$.*

iii. An action m is reachable in Σ *iff there exists a reachable step formed $a \xrightarrow{m} b$.*

Concurrent runs are now defined in two stages: Firstly, each action m is assigned an *action net*, representing the action's details in terms of an inscribed net. In a second step, those nets are "glued together", forming a concurrent run.

7.3 Definition *A state of a system net Σ is* contact free *iff for each $t \in T_\Sigma$ and each action m of t holds: if for each place $p \in {}^\bullet t$, $m(p,t) \subseteq a(p)$, then for each place $p \in t^\bullet$, $(m(t,p) \setminus m(p,t)) \subseteq A_p \setminus a(p)$.*

In the following we stick to system nets where each reachable state is contact free.

7.4 Definition *Let Σ be a system net, let $t \in T_\Sigma$, let m be an action of t, and let N be an injectively labeled net with $T_N = \{e\}$. Furthermore, assume $l(e) = (t,m)$, $l({}^\bullet e) = \{(p,a) \mid p \in {}^\bullet t$, and $a \in m(p,t)\}$, $l(e^\bullet) = \{(p,a) \mid p \in t^\bullet$, and $a \in m(t,p)\}$. Then N is an* action net of Σ *(for m).*

7.5 Definition *A net K is called an* occurrence net *iff*

i. for each $p \in P_K$, $|{}^\bullet p| \leq 1$ and $|p^\bullet| \leq 1$,

ii. for each $t \in T_K$, $|{}^\bullet t| \geq 1$ and $|t^\bullet| \geq 1$,

iii. the transitive closure F_K^+ of F_K, frequently written $<_K$, is irreflexive (i.e., $x_1 F_K x_2 F_K \ldots F_K x_n$ implies $x_1 \neq x_n$),

iv. for each $x \in K$, $\{y \mid y <_K x\}$ is finite.

7.6 Definition *Let Σ be a system net and let K be an element labeled occurrence net. K is a* concurrent run *of Σ iff*

i. in each concurrent state a of K, different elements of a are differently labeled,

ii. for each $t \in T_K$, $({}^\bullet t \cup t^\bullet, \{t\}, {}^\bullet t \times \{t\} \cup \{t\} \times t^\bullet)$ is an action net of Σ

iii. $l(\{p \in P_k \mid p^\bullet = \emptyset\})$ enables no action of Σ.

8 Structures and Terms

System nets have been represented in Sections 1-5 by means of *sorted terms*. Such terms ground on *structures*. This section provides the formal basis for structures and terms.

We first recall some basic notions on constants and functions:

8.1 Definition *Let $A_1, \ldots, A_k$ be sets.*

i. Let $a \in A_i$ for some $1 \leq i \leq k$. Then a is called a constant in the sets $A_1, \ldots, A_k$ *and A_i is called a* sort of a.

ii. For $i = 1, \ldots, n+1$ let $B_i \in \{A_1, \ldots, A_k\}$, and let $f : B_1 \times \ldots \times B_n \to B_{n+1}$ be a function. Then f is called a function over the sets $A_1, \ldots, A_k$. *The sets $B_1, \ldots, B_n$ are the* argument sorts *and B_{n+1} is the* target sort of f. *The $n+1$-tuple $(B_1, \ldots, B_{n+1})$ is the* arity of f *and is usually written $B_1 \times \ldots \times B_n \to B_{n+1}$.*

A *structure* is just a collection of constants and functions over some sets:

8.2 Definition *Let $A_1, \ldots, A_k$ be sets, let $a_1, \ldots, a_l$ be constants in $A_1, \ldots, A_k$ and let $f_1, \ldots, f_m$ be functions over $A_1, \ldots, A_k$. Then*

$$\mathcal{A} = (A_1, \ldots, A_k; a_1, \ldots, a_l; f_1, \ldots, f_m) \tag{78}$$

is a structure. *$A_1, \ldots, A_k$ are the* carrier sets, *$a_1, \ldots, a_l$ the* constants, *and $f_1, \ldots, f_m$ the* functions of $\mathcal{A}$.

The composition of functions of a structure can be described intuitively by means of *terms*. To this end, each constant a of a structure $\mathcal{A}$ is represented by a *constant symbol* $\mathtt{a}$ and likewise each function f of $\mathcal{A}$ by a *function symbol* $\mathtt{f}$. (This choice of symbols is just a matter of convenience and convention. Any other choice of symbols would do the same job). Furthermore, terms include *variables*:

8.3 Definition *Let $\mathcal{A} = (A_1, \ldots, A_k; a_1, \ldots, a_l; f_1, \ldots, f_m)$ be a structure.*

i. Let $X_1, \ldots, X_k$ be pairwise disjoint sets of symbols. For $x \in X_i$, call A_i the sort of x *$(i = 1, \ldots, k)$. Then $X = X_1 \cup \ldots \cup X_k$ is a set of* $\mathcal{A}$-sorted variables.

ii. Let X be a set of $\mathcal{A}$-sorted variables. For all $B \in \{A_1, \ldots, A_k\}$ we define the sets $T_B(X)$ of terms of sort B over X *inductively as follows:*

(a) $X_i \subseteq T_{A_i}$

(b) for all $1 \leq i \leq l$, if B is the sort of a_i then $\mathtt{a_i} \in T_B(X)$.

(c) For all $1 \leq i \leq m$, if $B_1 \times \ldots \times B_n \to B$ is the arity of f_i and if $t_j \in T_{B_j}(X)$ $(j = 1, \ldots, n)$ then $\mathtt{f}(t_1, \ldots, t_n) \in T_B(X)$.

iii. The set $T_{\mathcal{A}}(X) := T_{A_1}(X) \cup \ldots \cup T_{A_k}(X)$ is called the set of $\mathcal{A}$-terms over X.

In the following we always assume some (arbitrarily chosen, but) fixed *order* on variables. Generally we use the following notation:

A set M is said to be *ordered* if a unique tuple $(m_1, \ldots, m_k)$ of pairwise different elements m_i is assumed such that $M = \{m_1, \ldots, m_k\}$. We write $M = (m_1, \ldots, m_k)$ in this case.

Each term u over an ordered set of sorted variables describes a unique function, val^u, the *valuation* of u:

8.4 Definition *Let $\mathcal{A}$ be a structure and let $X = (x_1, \ldots, x_n)$ be an ordered set of $\mathcal{A}$-sorted variables. For $i = 1, \ldots, n$ let B_i be the sort of x_i and let $u \in T_B(X)$ for any sort B of $\mathcal{A}$. Then $B_1 \times \ldots \times B_n$ is the set of* arguments for X *and the* valuation of u in $\mathcal{A}$ *is a function $val^u : B_1 \times \ldots \times B_n \to B$, which is inductively defined over the structure of u:*

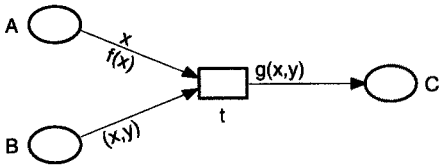
$$val^u(a_1, \ldots, a_n) = \begin{cases} a_i & \text{if } u = x_i \text{ for } 1 \leq i \leq n, \\ a & \text{if } u = \mathtt{a} \text{ for some constant } a \text{ of } \mathcal{A}, \\ f(val^{u_1}(a_1, \ldots, a_n), \ldots, val^{u_k}(a_1, \ldots, a_n)) & \\ & \text{if } u = \mathtt{f}(u_1, \ldots, u_k) \text{ for some function} \\ & f \text{ of } \mathcal{A} \text{ and terms } u_1, \ldots, u_k \in T_{\mathcal{A}}(X). \end{cases}$$

8.5 Definition *Let $\mathcal{A}$ be a structure.*

i. The set $T_{\mathcal{A}}(\emptyset)$ consists of the $\mathcal{A}$-ground terms *and is usually written $T_{\mathcal{A}}$.*
ii. For each $u \in T_{\mathcal{A}}$ of sort B, val^u is the unique function $val^u : \emptyset \to B$, i.e., val^u indicates a unique element in B. This element will be denoted val^u.

9 A Term Representation of System Nets

Based on structures and terms as introduced in the previous section, a representation of system nets is suggested in the following, as used in Sections 1-5. The representation of a transition's actions is the essential concept. To this end, each transition t is assigned its set M_t of *occurrence modes*. Each occurrence mode then defines an action. A typical example was



(79)

Assume the variable x is of sort M, y of sort N and x ordered before y. Then $M \times N$ is the set of occurrence modes of t. Each pair $(m, n) \in M \times N$ defines an action $\widetilde{mn}$ of t, gained by substituting m and n for x and y in the adjacent terms. Hence $\widetilde{mn}(A, t) = \{m, f(m)\}$, $\widetilde{mn}(B, t) = \{(m, n)\}$ and $\widetilde{mn}(t, C) = \{g(m, n)\}$.

The syntactical representation of term-based system nets reads as follows:

9.1 Definition *Let Σ be a net and let $\mathcal{A}$ be a structure. Assume*

i. each place $p \in P_\Sigma$ is assigned a carrier set A_p of $\mathcal{A}$ and a set $a_\Sigma(p) \subseteq T_{\mathcal{A}_p}$ of ground terms,
ii. each transition $t \in T_\Sigma$ is assigned an ordered set X_t of $\mathcal{A}$-sorted variables,
iii. each arc $f = (t, p)$ or $f = (p, t)$ adjacent to a transition t is assigned a set $\overline{f} \subseteq T_{\mathcal{A}_p}(X_t)$ of $\mathcal{A}_p$-terms over X_t.

Then Σ is called term inscribed over $\mathcal{A}$.

In graphical representations, the places p and the arcs (r, s) are inscribed by $a_\Sigma(p)$ and $\overline{rs}$, respectively. Occurrence modes and actions of a transition are defined as follows:

9.2 Definition *Let Σ be a term inscribed net and let $t \in T_\Sigma$ be a transition.*

i. Let $(x_1, \ldots, x_n)$ be the ordered set of variables of t and let M_i be the sort of x_i $(i = 1, \ldots, n)$. Then $M_t := M_1 \times \ldots \times M_n$ is the set of occurrence modes *of t.*

ii. Let $m \in M_t$. For each adjacent arc $f = (p, t)$ or $f = (t, p)$ and different $u, v \in \overline{f}$ assume $val^u(m) \neq val^v(m)$. Then $\widetilde{m}$ is an action of t, *defined by $\widetilde{m}(f) = \{val^u(m) \mid u \in \overline{f}\}$.*

The action $\widetilde{mn}$ discussed above is in fact an action of the transition (79). A term-inscribed net obviously represents a system net:

9.3 Definition *Let Σ be a net that is term-inscribed over a structure $\mathcal{A}$ such that for all $p \in P_\Sigma$ and all different $u, v \in a_\Sigma(p)$, $val^u \neq val^v$. Then the* system net of Σ *consists of*

- *the universe $\mathcal{A}$,*
- *for all $t \in T_\Sigma$, the actions of t as defined in Def. 9.2(ii),*
- *the initial state a, defined for each place $p \in P_\Sigma$ by $a(p) := \{val^u \mid u \in a_\Sigma(p)\}$.*

10 Set-Valued Terms

The formalism of Sect. 9 is adequate for many system nets. But there exist more general system nets requiring *set-valued* terms. In order to specify this issue more precisely, assume a system net Σ with a transition $t \in T_\Sigma$, an action m of t, and a place $p \in {}^\bullet t \cup t^\bullet$ with domain A. Then $\widetilde{m}(p, t)$ or $\widetilde{m}(t, p)$ is a subset of A, with each single term $u \in \overline{pt}$ or $u \in \overline{tp}$ contributing a single element, $val^u(m) \in A$. Now we suggest single terms v that contribute a *subset* $val^v(m) \subseteq A$. More precisely, *set-valued constant symbols*, *set-valued function symbols*, and *set-valued variables* will be used.

For the sake of uniform management of all cases, the evaluation $val^u(m)$ of terms u will be slightly adjusted, yielding a set $\mathrm{setval}^u(m)$ in any case:

10.1 Definition *Let Σ be a term inscribed net over a structure $\mathcal{A}$.*

i. Let $p \in P_\Sigma$ and let $u \in a_\Sigma(p)$. Then

$$\mathrm{setval}^u = \begin{cases} \{val^u\} & \text{if the sort of } u \text{ is } A_p \\ val^u & \text{if the sort of } u \text{ is } \mathcal{P}(A_p). \end{cases}$$

ii. Let $f = (p,t) \in F_\Sigma$ or $f = (t,p) \in F_\Sigma$, let $u \in \overline{f}$, and let m be an argument of X_t. Then

$$\mathrm{setval}^u(m) = \begin{cases} \{val^u(m)\} & \text{if the sort of } u \text{ is } A_p \\ val^u(m) & \text{if the sort of } u \text{ is } \mathcal{P}(A_p). \end{cases}$$

The actions of a term inscribed net with both element-valued and set-valued terms is now defined as follows:

10.2 Definition *Let Σ be a term inscribed net, let $t \in T_\Sigma$, and let $m \in M_t$ such that for each adjacent arc $f = (p,t)$ or $f = (t,p)$ and different $u, v \in \overline{f}$ we have $\mathrm{setval}^u(m) \cap \mathrm{setval}^v(m) = \emptyset$. Then $\widetilde{m}$ is an action of t, defined by $\widetilde{m}(f) = \bigcup_{u \in \overline{f}} \mathrm{setval}^u(m)$.*

10.3 Proposition *Let Σ be a term inscribed net, let $t \in T_\Sigma$, let m be an action of t, and let a be a state of Σ. For all $(r,s) \notin F_\Sigma$ let $\overline{rs} := \emptyset$.*

i. m is enabled at a iff, for each $p \in P_\Sigma$, $\bigcup_{u \in \overline{pt}} \mathrm{setval}^u(m) \subseteq a(p)$ and $(\bigcup_{u \in \overline{tp}} \mathrm{setval}^u(m) \setminus \bigcup_{u \in \overline{pt}} \mathrm{setval}^u(m)) \cap a(p) = \emptyset$.

ii. Let $a \xrightarrow{m} b$ be a step of Σ. Then for each $p \in P_\Sigma$, $b(p) = (a(p) \setminus \bigcup_{u \in \overline{pt}} \mathrm{setval}^u(m)) \cup \bigcup_{u \in \overline{tp}} \mathrm{setval}^u(m)$.

The system net of a term-inscribed net with both element-valued and set-valued terms is defined as a conservative extension of the corresponding notion in Sect. 9.3 for element-valued terms:

10.4 Definition *Let Σ be a net that is term-inscribed over a structure $\mathcal{A}$, such that for all $p \in P_\Sigma$ and all different $u, v \in a_\Sigma(p)$ holds $\mathrm{setval}^u \cap \mathrm{setval}^v = \emptyset$. Then the system net of Σ consists of*

- *the universe of $\mathcal{A}$,*
- *for all $t \in T_\Sigma$, the actions of t as defined in Def. 10.2,*
- *the initial state a, defined for each place $p \in P_\Sigma$ by $a(p) := \bigcup_{u \in a_\Sigma(p)} \mathrm{setval}^u$.*

We are now prepared to define *schemata* for system nets: a *system schema* is a term-inscribed net with the underlying structure not entirely fixed. Thus, a system schema represents a *set* of system nets. A representation of a system schema declares some *sorts* (domains) and some constants, functions, and variables over standard sorts, declared sorts, cartesian products, or powersets of sorts. We furthermore assume standard sorts such as the natural numbers *nat* or the truth values *bool*, together with the usual operations. Some additional requirements may focus the intended interpretations.

The distributed algorithms of Chapters 1-5 are all represented as system schemata. This is crucial, as each distributed algorithm is to run on *any* network out of a class of networks. Each interpretation of the involved symbols then yields one concrete network.

11 First-Order State Properties

Properties of system nets and system schemata are represented in a logical framework. Terms as introduced in Sect. 9 (there used as arc inscriptions) will serve in a first-order logic, with places of system nets as predicate symbols.

We start with the syntax of formulas over a structure $\mathcal{A}$.

11.1 Definition *Let $\mathcal{A}$ be a structure, let X be a set of $\mathcal{A}$-sorted variables, and let P be any set of symbols. Then the set $\mathcal{F}(\mathcal{A}, X, P)$ of* state formulas over $\mathcal{A}$, X, and P *is the smallest set of symbol chains such that for all $t \in T_{\mathcal{A}}(X)$ and all $p, q \in P$,*

i. $p.t$, $p = t$, *and* $p \subseteq q \in \mathcal{F}(\mathcal{A}, X, P)$
ii. if $f, g \in \mathcal{F}(\mathcal{A}, X, P)$ *then* $f \wedge g \in \mathcal{F}(\mathcal{A}, X, P)$ *and* $\neg f \in \mathcal{F}(\mathcal{A}, X, P)$.

In the sequel we employ the conventional propositional symbols $\vee$ and $\rightarrow$, and for any set $Q = \{q_1, \ldots, q_n\}$ the shorthands $\bigvee Q$ for $q_1 \vee \ldots \vee q_n$, and $\bigwedge Q$ or just Q for $q_1 \wedge \ldots \wedge q_n$. Furthermore, we write $A.u_1, \ldots, u_n$ as a shorthand for $A.u_1 \wedge \ldots \wedge A.u_n$.

Each system schema Σ is assigned its set of state formulas. Those formulas are constructed from the structure of Σ, with the places of Σ serving as predicate symbols. The token load $s(p)$ of place p at a state s, as well as the inscriptions in $\overline{f}$ of an arc f, are terms that may occur in state formulas.

11.2 Definition *Let $\mathcal{A}$ be a structure, let X be an $\mathcal{A}$-sorted set of variables, and let Σ be a net, term-inscribed over $\mathcal{A}$ and X.*

i. Each $f \in \mathcal{F}(\mathcal{A}, X, P_\Sigma)$ is a state formula of Σ.
ii. For each state s of Σ, the state formula $\hat{s}$ of Σ is defined by $\hat{s} := \bigwedge_{p \in s} p \wedge \bigwedge_{p \notin s} \neg p$.

Such formulas are interpreted as follows:

11.3 Definition *Let Σ be a net, let f be a state formula of Σ, let v be an argument for its variables, and let s be a state of Σ.*

i. $s, v \models f$ is inductively defined over the structure of f. To this end, let $u \in T_{\mathcal{A}}(X)$, $p, q \in P_\Sigma$ and $g, h \in \mathcal{F}(\mathcal{A}, X, P)$.
  - *$s, v \models p.t$ iff $\mathrm{setval}^u(v) \subseteq s(p)$, and*
    $s, v \models p = t$ iff $\mathrm{setval}^u(v) = s(p)$.
  - *$s, v \models p \subseteq q$ iff $s(p) \subseteq s(q)$.*
  - *$s, v \models g \wedge h$ iff $s, v \models g$ and $s, v \models h$.*
  - *$s, v \models \neg g$ iff not $s, v \models g$.*

ii. $s \models f$ iff, for all arguments u of X, $s, u \models f$.
iii. $\Sigma \models f$ iff, for all reachable states s of Σ, $s \models f$.

Apparently, for each state a, $a \models \hat{a}$.

12 Multisets and Linear Functions

State properties can frequently be proven by means of equations and inequalities, which in turn can be derived from the static structure of a given system net. Each place of the net will serve as a variable, ranging over the subsets of the places' domains.

Each structure $\mathcal{A}$ canonically induces *multisets* of its carrier sets and linear extensions of its functions. Intuitively, a multiset B over a set A assigns to each $a \in A$ a multiplicity of occurrences of a. As a special case, a conventional subset of a sticks to the multiplicities 0 and 1. For technical convenience we allow negative multiplicities, too, but *proper* multisets have no negative entry.

12.1 Definition *Let A be a set.*

i. Any function $M : A \to \mathbb{Z}$ is called a multiset *over A. Let $A^{\mathfrak{M}}$ denote the set of all multisets over A.*
ii. Let $M \in A^{\mathfrak{M}}$ and $z \in \mathbb{Z}$. Then $zM \in A^{\mathfrak{M}}$ is defined for each $a \in A$ by $zM(a) := z \cdot M(a)$.
iii. Let $L, M \in A^{\mathfrak{M}}$. Then $L + M \in A^{\mathfrak{M}}$ is defined for each $a \in A$ by $(L + M)(a) := L(a) + M(a)$.
iv. A multiset $M \in A^{\mathfrak{M}}$ is proper *iff $M(a) \geq 0$ for all $a \in A$.*

Sets can be embedded canonically into multisets.

12.2 Definition *Let A be a set, let $a \in A$ and $B \subseteq A$. If A is obvious from the context, a^m and B^m denote multisets over A, defined by $a^m(x) = 1$ if $x = a$ and $a^m(x) = 0$ otherwise; and $B^m(x) = 1$ if $x \in B$ and $B^m(x) = 0$, otherwise.*

By abuse of notation we usually write just A instead of A^m.

There is a canonically defined scalar product and a sum of functions over multisets:

12.3 Definition *Let A and B be sets.*

i. Any function $\varphi : A^{\mathfrak{M}} \to B^{\mathfrak{M}}$ is called a multiset function from A to B.
ii. Let $\varphi : A^{\mathfrak{M}} \to B^{\mathfrak{M}}$ be a multiset function and let $z \in \mathbb{Z}$. Then $z\varphi : A^{\mathfrak{M}} \to B^{\mathfrak{M}}$ is defined for each $M \in A^{\mathfrak{M}}$ by $z\varphi(M) := z \cdot (\varphi(M))$.
iii. Let $\varphi, \psi : A^{\mathfrak{M}} \to B^{\mathfrak{M}}$ be two multiset functions. Then $\varphi + \psi : A^{\mathfrak{M}} \to B^{\mathfrak{M}}$ is defined for each $M \in A^{\mathfrak{M}}$ by $(\varphi + \psi)(M) := \varphi(M) + \psi(M)$.
iv. $\mathbb{O}_{AB}$ denotes the zero-valuating multiset function from A to B, i.e., $\mathbb{O}_{AB}(M) = \mathbb{O}_B$ for each $M \in A^{\mathfrak{M}}$. The index AB is skipped whenever it can be assumed from the context.

Each function $f : A \to B$ and each set-valued function $g : A \to B^{\mathfrak{M}}$ of a structure $\mathcal{A}$ can be extended canonically to a multiset function $g : A^{\mathfrak{M}} \to B^{\mathfrak{M}}$:

12.4 Definition *Let A and B be sets and let $f : A \to B$ or $f : A \to B^{\mathfrak{M}}$ be a function. Then the multiset function $\hat{f} : A^{\mathfrak{M}} \to B^{\mathfrak{M}}$ is defined for each $M \in A^{\mathfrak{M}}$ and each $b \in B$ by $\hat{f}(m)(b) = \Sigma_{a \in f^{-1}(b)} M(a)$.*

By abuse of notation we write f instead of $\hat{f}$ whenever the context excludes confusion. The induced functions $\hat{f}$ are *linear*:

12.5 Lemma *Let A and B be sets, let $f : A \to B$ be a function, let $L, M \in \mathfrak{M}(A)$, and let $z \in \mathbb{Z}$. Then for the multiset extension of f, $\hat{f}(L+M) = \hat{f}(L) + \hat{f}(M)$, and $\hat{f}(z \cdot M) = z \cdot \hat{f}(M)$.*

13 Place Weights, System Equations, and System Inequalities

State properties are essentially based on weighted sets of tokens, formally given by multiset valued mappings on the places' domains.

13.1 Definition *Let Σ be a system net over a universe $\mathcal{A}$, let $p \in P_\Sigma$, and let B be any multiset. Then a mapping $I : \mathcal{A}_p \to B$ is a* place weight of p. *I is* natural *if $B = \mathbb{N}$.*

Place weights are frequently extended to set-valued arguments and then applied to the token load $s(p)$ of the token at place p in a global state, s. In this case, a multiset $I(s(p))$ is called a *weighted token load of p.*

Place weights can be used to describe invariant properties of system nets by help of equations that hold in all reachable states:

13.2 Definition *Let Σ be a system net over a universe $\mathcal{A}$, let B be any multiset and let $P = \{p_1, \ldots, p_n\} \subseteq P_\Sigma$. For $j = 1, \ldots, k$, let $I^j : \mathcal{A}_{p_j} \to B$ be a place weight of p_j.*

i. $\{I^1, \ldots, I^k\}$ *is a* Σ-invariance with value B *if for each reachable state s of Σ,*

$$I^1(s(p_1)) + \cdots + I^k(s(p_k)) = B.$$

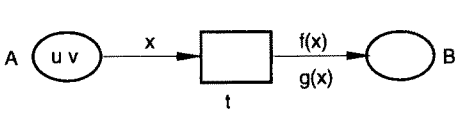
ii. A Σ-invariance $\{I^1, \ldots, I^k\}$ is frequently written as a symbolic equation

$$I^1(\mathrm{p_1}) + \cdots + I^k(\mathrm{p_k}) = B$$

and this equation is said to hold in Σ.

In a Σ-equation $I^1(\mathrm{p_1}) + \cdots + I^k(\mathrm{p_k}) = B$, the value of B is apparently equal to $I^1(s_\Sigma(p_1)) + \cdots + I^k(s_\Sigma(p_k))$, with s_Σ the initial state of Σ.

As a technical example, in the term inscribed representation of a system net Σ,



sort	dom	
const	u, v : dom	
fct	f, g : dom → dom	(80)
var	x : dom	

let $\{u, v\}$ be the domain of both A and B, and for $x \in \{u, v\}$ let $I^A(x) = f(x) + g(x)$ and $I^B(x) = x$. Then $\{I^A, I^B\}$ is a Σ-invariance with value $U = f(u) + g(u) + f(v) + g(v)$, symbolically written

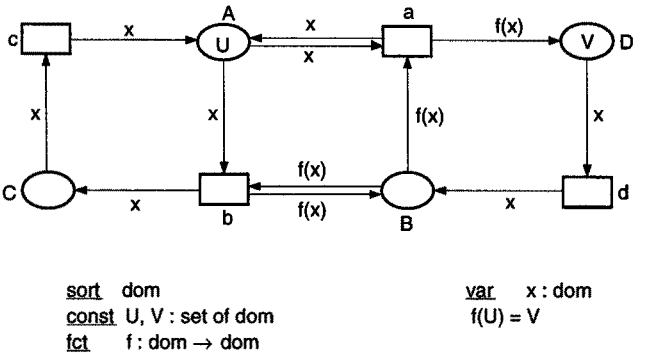


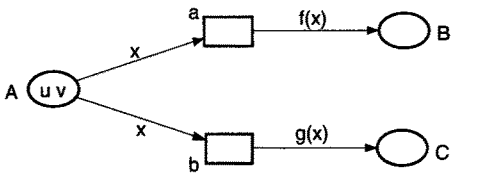
Figure 13.1. $f(A) + B \geq V$ is a valid inequality

$$f(A) + g(A) + B = U. \tag{81}$$

One of the reachable states is s, with $s(A) = u$ and $s(B) = f(v) + g(v)$. Then in fact $I^A(s(A)) + I^B(s(B)) = I^A(u) + I^B(f(v)) + I^B(g(v)) = U$.

Intuitively formulated, according to this invariance, the element u is at A, or both $f(u)$ and $g(u)$ are at B. The corresponding property for v holds accordingly in Σ.

As a further example, in $\Sigma =$



(82)

let again $\{u, v\}$ be the domain of all places A, B, and C, and for $x \in \{u, v\}$ let $I^A(x) = x$, $I^B(x) = f^{-1}(x)$ and $I^C(x) = g^{-1}(x)$. Then $\{I^A, I^B, I^C\}$ is a Σ-invariance with value $u + v$, symbolically written $A + f^{-1}(B) + g^{-1}(C) = u + v$. One of the reachable states is s, with $s(A) = u$, $s(B) = f(v)$ and $s(C) = \emptyset$. Then in fact $I^A(s(A)) + I^B(s(B)) + I^C(s(C)) = I^A(u) + I^B(f(v)) + I^C(\emptyset) = u + f^{-1}(f(v)) = u + v$.

13.3 Definition *Let Σ be a system net over a universe $\mathcal{A}$, let B be any multiset, and let $P = \{p_1, \ldots, p_k\} \subseteq P_\Sigma$. For $j = 1, \ldots, k$ let $I^j : \mathcal{A}_{p_j} \to B$ be a place weight of p.*

$\{I^1, \ldots, I^k\}$ yields a Σ-socket with value B *if for each reachable state s of Σ,*

$$I^1(s(p_1)) + \cdots + I^k(s(p_k)) \geq B.$$

A Σ-socket $\{I^1, \ldots, I^k\}$ is frequently written as a symbolic inequality

$$I^1(\mathrm{p_1}) + \cdots + I^k(\mathrm{p_k}) \geq B,$$

and this inequality is said to hold in Σ.

Figure 13.1 provides a typical example.

In $\Sigma_{13.1}$ let I^A and I^B be place weights of A and B, respectively, with $I^A(x) = f(x)$ for each $x \in U$ and $I^B(y) = y$ for each $y \in V$. Then $\{I^A, I^B\}$ is a Σ-socket with value V. As a symbolic inequality it reads $f(A) + B \geq V$.

14 Place Invariants of System Nets

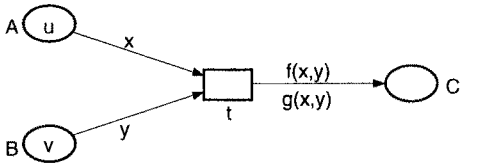
We are now seeking a technique to prove Σ-invariances without explicitly visiting all reachable states. To this end we construct *place invariants*: a set of place weights is a place invariant if each occurrence mode m of each transition t yields a balanced weighted effect to the places involved, i.e., the weighted set of removed tokens is equal to the weighted set of augmented tokens; formally, for place weights $I^1, \ldots, I^k$ of places $p_1, \ldots, p_k$,

$$I^1(m(t,p_1)) + \cdots + I^k(m(t,p_k)) = I^1(m(p_1,t)) + \cdots + I^k(m(p_k,t)). \tag{84}$$

A more concise representation of (84) is gained by a slightly different perspective on transitions and their actions: Each arc $\beta = (p,t)$ or $\beta = (t,p)$ defines a mapping $\widetilde{\beta}$ that assigns each action m of t the corresponding subset $m(\beta)$ of A_p. Furthermore, this subset is canonically conceived as a multiset, i.e., an element of $A_p^{\mathfrak{M}}$:

14.1 Definition *Let Σ be a system over a structure $\mathcal{A}$. Let $t \in T_\Sigma$ be a transition with M_t its set of actions and let $\beta = (t,p)$ or $\beta = (p,t)$ be an arc of Σ. Then the function $\widetilde{\beta} : M_t \to A_p^{\mathfrak{M}}$ is defined by $\widetilde{\beta}(m) = m(\beta)$.*

The function $\widetilde{\beta}$ is canonically extended to $\widetilde{\beta}(m) = \emptyset$ if β is no arc. For example, in




```
sort  U, V, W
const u : U
const v : V
fct   f, g : U × V → W
var   x : U
var   y : V
```
(85)

the set of actions of t is $U \times V$. Then each action (u,v) yields

$$\widetilde{At}(u,v) = \{u\},\ \widetilde{Bt}(u,v) = \{v\},\ \widetilde{tC}(u,v) = \{f(u,v), g(u,v)\}, \text{ and}$$
$$\widetilde{tA}(u,v) = \widetilde{tB}(u,v) = \widetilde{Ct}(u,v) = \emptyset. \tag{86}$$

$\widetilde{tp} - \widetilde{pt}$ is a multiset valued function that assigns each occurrence mode m of t its effect on p, i.e., the tokens removed from p or augmented to p upon t's occurrence in mode m.

Each place weight $I^p : A_p \to B$ of a place p can canonically be extended to the set valued arguments $I^p : A_p^{\mathfrak{M}} \to B^{\mathfrak{M}}$, by Def. 12.4. This function in turn can be composed with $\widetilde{tp} - \widetilde{pt}$, yielding a function $I^p \circ (\widetilde{tp} - \widetilde{pt}) : M_t \to B^{\mathfrak{M}}$.

A set of place weights is a *place invariant* if the sum of weighted effects of all involved places reduces to the zero function $\mathbb{0}$. The *value* of a place invariant is derived from the net's initial state:

14.2 Definition *Let Σ be a system net and let $p_1, \ldots, p_k \in P_\Sigma$. For $j = 1, \ldots, k$ let I^j be a place weight of p_j. Then $I = \{I^1, \ldots, I^k\}$ is a* place invariant of Σ *if for each transition $t \in T_\Sigma$,*

$$I^1 \circ (\widetilde{tp_1} - \widetilde{p_1 t}) + \cdots + I^k \circ (\widetilde{tp_k} - \widetilde{p_k t}) = \mathbb{O}.$$

The multiset $I^1(s_\Sigma(p_1)) + \cdots + I^k(s_\Sigma(p_k))$ is the value of I.

A place invariant provides in fact a valid Σ-equation:

14.3 Theorem *Let Σ be a system net, let $p_1, \ldots, p_k \in P_\Sigma$, and for $j = 1, \ldots, k$, let I^j be a place weight of Σ. Let $\{I^1, \ldots, I^k\}$ be a place invariant of Σ and let U be its value. Then the equation*

$$I^1(p_1) + \cdots + I^k(p_k) = U$$

holds in Σ.

Place invariants can be mimicked symbolically in term-inscribed representations of system nets. To this end, the functions $\widetilde{tp}$, $\widetilde{pt}$, $\widetilde{tp} - \widetilde{pt}$, and I^p will be represented symbolically. The composition $I^p \circ (\widetilde{tp} - \widetilde{pt})$ of functions I^p and $(\widetilde{tp} - \widetilde{pt})$ then is symbolically executable as substitution of terms.

Definition 9.1 assigns each arc $\beta = (t, p)$ or $\beta = (p, t)$ of a term-inscribed net Σ a set $\overline{\beta} \subseteq T_{A_p}(X_t)$ of A_p-terms over X_t. For each $u \in \overline{\beta}$, val^u (as defined in Def. 8.4) is a mapping from M_t to $\mathcal{A}_p$. This mapping can be extended canonically to $val^u : M_t \to \mathcal{A}_p^{\mathfrak{M}}$. Mappings of this kind can be summed up, giving rise to the mapping $\widetilde{\beta} : M_t \to \mathcal{A}_p^{\mathfrak{M}}$ of Def. 14.1, defined by $\widetilde{\beta}(m) := val^{u_1}(m) + \cdots + val^{u_k}(m)$, with $X_t = \{u_1, \ldots, u_k\}$. Hence $\widetilde{\beta}$ can be represented symbolically as

$$\widetilde{\beta} = u_1 + \cdots + u_k \tag{87}$$

in this case.

The multiset extension $I^p : \mathcal{A}_p^{\mathfrak{M}} \to B$ of a place weight $I : A_p \to B$ can be represented as a term with one variable, ranging over $\mathcal{A}_p^{\mathfrak{M}}$. For the sake of convenience we always choose the variable p, hence the corresponding term is an element of $T_B(\{\mathrm{p}\})$.

The composed function $I^p \circ (\widetilde{tp} - \widetilde{pt}) : M_t \to B$ is now symbolically represented by the multiset term

$$\tau = I^p[\widetilde{tp} - \widetilde{pt}/\mathrm{p}] \tag{88}$$

which is gained from I^p by replacing each occurrence of the variable p in I^p by the term $\widetilde{tp} - \widetilde{pt}$. Hence τ is a term in $T_B(X_t)$, and its valuation val^τ is equal to $I^p \circ (\widetilde{tp} - \widetilde{pt})$.

15 Traps of System Nets

We are now seeking a technique to prove Σ-sockets without visiting all reachable states. To this end we construct *initialized traps* for system nets, in analogy to initialized traps of elementary system nets.

Informally stated, a trap of a system net is a set $\{I^1, \ldots, I^k\}$ of weights of places $p_1, \ldots, p_k$ such that for each element b of a given set B, each transition that removes at least one token with weight b from those places returns at least one token with weight b to those places. This gives rise to an inequality of the form

$$I^1(p_1) + \cdots + I^k(p_k) \geq B. \tag{89}$$

Traps are essentially a matter of plain sets (whereas place invariants are based on multisets). For an arc (p,t) and an occurrence mode m of t, $m(p,t)$ is a plain set according to Def. 6.2. Then $I(m(p,t)) := \{I(u) \mid u \in m(p,t)\}$ is a set, for any place weight I. Therefore, the definition of traps goes with set union (not with multiset addition).

15.1 Definition *Let Σ be a system net and let $p_1, \ldots, p_k \in P_\Sigma$. For $j = 1, \ldots, k$, let I^j be a place weight of p_j. Then $I = \{I^1, \ldots, I^k\}$ is a* trap *of Σ if for each transition $t \in T_\Sigma$ and each occurrence mode m,*

$$I^1(m(p_1,t)) \cup \cdots \cup I^k(m(p_k,t)) \subseteq I^1(m(t,p_1)) \cup \cdots \cup I^k(m(t,p_k)).$$

The set $I^1(s_\Sigma(p_1)) \cup \cdots \cup I^k(s_\Sigma(p_k))$ is the initialization *of I.*

. An initialized trap in fact provides a valid Σ-inequality:

15.2 Theorem *Let Σ be a system net, let $p_1, \ldots, p_k \in P_\Sigma$, and for $j = 1, \ldots, k$, let I^j be a place weight of Σ. Let $\{I^1, \ldots, I^k\}$ be a trap of Σ with initialization B. Then the inequality*

$$I^1(p_1) \cup \cdots \cup I^k(p_k) \geq B$$

holds in Σ.

Proof of traps can be mimicked symbolically in term-inscribed system nets. To this end, place weights I, and functions $\widetilde{\beta}$ assigned to arcs β, are represented symbolically as described in Sect. 14. The function $I \circ \beta$ can then be represented symbolically by the multiset term

$$\tau = I^p[\widetilde{\beta}/\mathrm{p}] \tag{90}$$

in analogy to (88) of Sect. 14. Union of functions then can be expressed by set union of singleton sets $\{\tau\}$. Each valuation of the variable p in τ by some $m \in \mathcal{A}_p$ then describes the item $I^p \circ \widetilde{\beta}(m) = I^p(\widetilde{\beta}(m))$.

16 Progress on Interleaved Runs

A progress property $p \mapsto q$ (p *leads to* q) is constructed from two state properties p and q. $p \mapsto q$ holds in an interleaved run w if each p-state of w is followed by a q-state. $p \mapsto q$ holds in a system net Σ if $p \mapsto q$ holds in each of its interleaved runs. Technically, leads-to formulas are constructed from state formulas:

16.1 Definition *Let $\mathcal{A}$ be a structure, let X be a set of $\mathcal{A}$-sorted variables, let P be a set of symbols, and let $p, q \in \mathcal{F}(\mathcal{A}, X, P)$ be state formulas. Then the symbol sequence $p \mapsto q$ (p leads to q) is a* first-order leads-to formula.

Leads-to formulas are interpreted over interleaved runs and over system nets:

16.2 Definition *Let Σ be a net that is term-inscribed over a structure $\mathcal{A}$ and a set X of variables. Let $p, q \in \mathcal{F}(\mathcal{A}, X, P_\Sigma)$ and let w be an interleaved run of Σ.*

i. For an argument u of X let $w \models (p \mapsto q)(u)$ iff for each $p(u)$-state with index i, there exists a $q(u)$-state with index $j \geq i$.
ii. $p \mapsto q$ is said to hold in *w (written $w \models p \mapsto q$) iff for each argument u of X, $w \models (p \mapsto q)(u)$.*
iii. $p \mapsto q$ is said to hold in *Σ (written $\Sigma \models p \mapsto q$) iff $w \models p \mapsto q$ for each interleaved run w of Σ.*

16.3 Definition *Let Σ be a system net and let s be a state of Σ.*

i. s is progress prone *iff s enables at least one action.*
ii. Let $t \in T_\Sigma$ and let m be an action of t. s prevents *m iff there exists some place p of Σ, such that $\Sigma \models \hat{s} \to \neg m(p, t)$.*
iii. Let $t \in T_\Sigma$ and let m be an action of t. $m \in s^\bullet$ if for some place p of Σ, $s(p) \cap m(p, t) \neq \emptyset$.
iv. A set M of actions of some transitions of Σ is a change set of *s if $M \neq \emptyset$ and s prevents each $m \in s^\bullet \setminus M$.*

The following theorem describes the most general case for picking up leads-to formulas from the static structure of a system net: Each change set of a progress prone state s yields a leads-to formula:

16.4 Theorem *Let Σ be a system net, let s be a progress prone state, and let M be a change set of s. Then*

$$\Sigma \models s \mapsto \bigvee_{m \in M} \mathrm{eff}(s, m).$$

17 Progress of Concurrent Runs

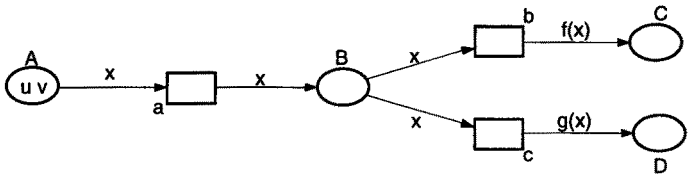
17.1 Definition *Let $\mathcal{A}$ be a structure, let X be a set of $\mathcal{A}$-sorted variables, let P be a set of symbols, and let $p, q \in \mathcal{F}(\mathcal{A}, X, P)$ be state formulas. Then the symbol sequence $p \hookrightarrow q$ ("p causes q") is a* first-order causes formula.

Causes formulas are interpreted over concurrent runs and over system nets:

17.2 Definition *Let Σ be a net that is term-inscribed over a structure $\mathcal{A}$ and a set X of variables. Let $p, q \in \mathcal{F}(\mathcal{A}, X, P_\Sigma)$ and let K be a concurrent run of Σ.*

i. For an argument u of X, let $K \models (p \hookrightarrow q)(u)$ iff to each reachable $p(u)$-state C of K there exists a $q(u)$-state D of K that is reachable from C.
ii. $p \hookrightarrow q$ is said to hold in *K (written $K \models p \hookrightarrow q$) iff for each argument u of X, $K \models (p \hookrightarrow q)(u)$.*
iii. $p \hookrightarrow q$ is said to hold in *Σ (written $\Sigma \models p \hookrightarrow q$) iff $K \models p \hookrightarrow q$ for each concurrent run K of Σ.*

As an example, $A.\{u,v\} \hookrightarrow B.\{u,v\}$ holds in



(91)

17.3 Lemma *Let Σ be a system net that is term-inscribed over a structure $\mathcal{A}$ and let $p, q \in \mathcal{F}(\mathcal{A}, X, P_\Sigma)$.*

i. $\Sigma \models p \hookrightarrow p$.
ii. If $\Sigma \models p \hookrightarrow q$ and $\Sigma \models q \hookrightarrow r$ then $\Sigma \models p \hookrightarrow r$.
iii. If $\Sigma \models p \hookrightarrow r$ and $\Sigma \models q \hookrightarrow r$ then $\Sigma \models (p \vee q) \hookrightarrow r$.
iv. If $\Sigma \models p \mapsto q$ then $\Sigma \models p \hookrightarrow q$.
v. If q includes no logical operator and $\Sigma \models p \hookrightarrow q$ then $\Sigma \models p \mapsto q$.

A rule to pick up *causes* properties from a system net is now derived, in an entirely *semantical* framework.

We start with some properties and notations of states of system nets.

17.4 Definition *Let Σ be a system net and let r, s be two states of Σ.*

i. The state $r \cup s$ of Σ is defined for each place $p \in P_\Sigma$ by $(r \cup s)(p) := r(p) \cup s(p)$.
ii. Let $r \subseteq s$ iff for each place $p \in P_\Sigma$, $r(p) \subseteq s(p)$.
iii. r is disjoint with *s iff for each $p \in P_\Sigma$, $r(p) \cap s(p) = \emptyset$.*
iv. For an action m of some transition t, let ${}^\bullet m$ be a state of Σ, defined for each place $p \in P_\Sigma$ by ${}^\bullet m(p) = m(p,t)$. For a set M of actions, let ${}^\bullet M$ be the state defined for each $p \in P_\Sigma$ by ${}^\bullet M(p) = \bigcup\{m(p) \mid m \in M\}$.

Change sets of system nets, as defined in Def. 16.3 for interleaved progress, can likewise be used for concurrent progress properties:

17.5 Theorem *Let Σ be a system net and let r, s be states of Σ. Assume s is progress prone, and let $U = V \cup W$ be a change set of s, with ${}^\bullet V \subseteq s$ and r disjoint with ${}^\bullet V$. Then $\Sigma \models r \cup s \hookrightarrow (r \cup \bigvee_{u \in V} \mathrm{eff}(s,u)) \vee (\bigvee_{u \in W} \mathrm{eff}(r \cup s, u))$.*

Many applications of this theorem deal with the special case of $W = \emptyset$, i.e., ${}^\bullet U \subseteq s$ and r disjoint with s:

17.6 Corollary *Let Σ be a system net, let s be a progress prone state of Σ, and let U be a change set of s with ${}^\bullet U \subseteq s$. Furthermore, let r be a state that is disjoint with s. Then $\Sigma \models r \cup s \hookrightarrow r \cup (\bigvee_{u \in U} \mathrm{eff}(s,u))$.*

18 Ground Formulas and Rounds

18.1 Definition *Let Σ be a system net and let p be a state formula of Σ. Then p is a* ground formula of Σ *if* $\Sigma \models \text{true} \hookrightarrow p$.

18.2 Theorem *Let Σ be a system net and let s be a state of Σ. Then s is a ground formula of Σ iff $\Sigma \models a_\Sigma \hookrightarrow s$ and for each element u of some change set U holds: $\Sigma \models \text{eff}(s,u) \hookrightarrow s$.*

18.3 Theorem *Let Σ be a system and let p be a ground formula of Σ. Let s be a state of Σ with $\Sigma \models s \to \neg p$, and let U be a change set of s. Then $\Sigma \models s \hookrightarrow \bigvee_{u \in U} \text{eff}(s,u)$.*

References

[Agha 86] G. A. Agha: *A Model of Concurrent Computation in Distributed Systems.* MIT Press, Cambridge, MA (1986)

[Alpern, Schneider 85] Bowen Alpern, Fred B. Schneider: *Defining Liveness.* Information Processing Letters 21, pp. 181–185 (1985)

[Ben-Ari 90] M. Ben-Ari: *Principles of Concurrent and Distributed Programming.* International Series in Computer Science. Prentice Hall, Englewood Cliffs, N. J., (1990)

[Barbosa 96] V. Barbosa: *An Introduction to Distributed Algorithms.* MIT Press, Cambridge, MA (1996)

[Berry, Boudol 82] G. Berry, G. Boudol: *The chemical abstract machine.* TCS, (1982)

[Banâtre, Coutant, le Metayer 88] J.-P. Banâtre, A. Coutant, D. le Metayer: *A Parallel Machine for Multiset Transformation and its Programming Style.* Future Generations Computer Systems 4, pp. 133–144 (1988)

[Burns, Esparza 96] G. Burns, J. Esparza: *Trapping mutual exclusion in the box calculus.* Theoretical Computer Science. Special Volume on Petri Nets 153, (1 2), (January 1996)

[Bennett 73] C. H. Bennett: *Logical Reversibility of Computation.* IBM Journal of Research and Development 6, pp. 525–532 (1973)

[Best 96] Eike Best: *Semantics of Sequential and Parallel Programs* International Series in Computer Science. Prentice Hall, Englewood Cliffs, N. J. , (1996)

[Best, Fernandez 88] Eike Best, Cèsar Fernandez: *Nonsequential Processes.* volume 13 of EATCS Monographs on Theoretical Computer Science, Springer-Verlag, Berlin, (1988)

[Brown, Gouda, Wu 89] G. M. Brown, M. G. Gouda, C. Wu: *Token systems that self-stabilize.* IEEE Transaction on Computers 38(6), pp. 845–852 (1989)

[Burnes, Pachl 89] J. E. Burnes, J. Pachl: *Uniform self-stabilizing rings.* ACM Transactions on Programming Languages and Systems 11(2), pp. 330–344 (April 1989)

[Broy 87] Manfred Broy: *Semantics of Finite and Infinite Networks of Concurrent Communicating Agents.* Distributed Computing 2, pp. 13–31 (1987)

[Chang 82] E. J. H. Chang: *Echo algorithms: Depth parallel operations on general graphs.* IEEE Transactions on Software Engineering SE-8(4), pp. 391–401 (1982)

[Chandy, Misra 84] K. M. Chandy, J. Misra: *The drinking philosophers problem.* ACM Transactions on Programming Languages and Systems 6(4), pp. 632–646 (October 1984)

[Chandy, Misra 88] K. M. Chandy, J. Misra: *Parallel Program Design: A foundation.* Addison-Wesley, MA (1988)

[Desel 97] J. Desel: *How distributed algorithms play the token game.* In: C. Freksa, M. Jantzen, R. Valk (eds.): Foundations of Computer Science, Vol 1337 LNCS Lecture Notes in Computer Science, pp. 297–306 Springer-Verlag, Berlin (1997)

[Dijkstra 71] Edsger W. Dijkstra: *Hierarchical ordering of sequential processes.* Acta Informatica 1, pp. 115–138 (1971)

[Dijkstra 74] E. W. Dijkstra: *Self-stabilizing systems in spite of distributed control.* Communications of the ACM 17(11), pp. 643–644 (1974)

[Dijkstra 75] E. W. Dijkstra: *Guarded commands, nondeterminancy, and formal derivation of programs.* Comm. ACM 18(8), pp. 453–457 (1975)

[Dijkstra 78] E. W. Dijkstra: *Finding the correctness proof of a concurrent program.* Proc. Koninklijke Nederlandse Akademie van Wetenschappen 81(2), pp. 207–215 (1978)

[Desel, Kindler 98] J. Desel, E. Kindler: *Proving correctness of distributed algorithms using high-level Petri nets - a case study.* In: 1998 International Conference on Application of Concurrency to System Design, pp. 177–186 Fukushima, Japan, (March 1998) IEEE Computer Society Press.

[Desel, Kindler, Vesper, Walter 95] J. Desel, E. Kindler, T. Vesper, R. Walter: *A simplified proof for a self-stabilizing protocol: A game of cards.* Information Processing Letters 54, pp. 327–328 (1995)

[Desel, Kindler, Walter 94] J. Desel, E. Kindler, R. Walter: *A game of tokens: A proof contest.* Petri Net Newsletter 47, pp. 3–4 (October 1994)

[Dijkstra, Scholten 80] E. W. Dijkstra, C. S. Scholten: *Termination Detection for Diffusing Computations.* Inf. Proc. Letters 4,pp. 1-4 (1980)

[Finn 79] S. G. Finn: *Resynch procedures and a fail safe network protocol.* IEEE Transactions on Communications, COM 27, pp. 840–845 (1979)

[Fredkin, Toffoli 82] Edward Fredkin, Tommaso Toffoli: *Conservative Logic.* International Journal of Theoretical Physics, Vol. 21, Nos. 3/4, pp. 219–253 (1982)

[Gandy 80] R. Gandy: *Church's Thesis and Principles for Mechanisms.* In: J. Barwise et al (eds.): "The Kleene Symposium", North-Holland, Amsterdamm, pp. 123–148 (1980)

[Gehrke, Plaxton, Rajaraman 97] J. E. Gehrke, C. G. Plaxton, R. Rajaraman: *Rapid convergence of a local load balancing algorithm for asynchronous rings.* In: M. Mavronicolas, P. Tsigas (eds.): "Distributed Algorithms, WDAG", Vol. 1320 of: LNCS Lecture Notes in Computer Science, pp. 81–95 Springer-Verlag (September 1997)

[Harel 87] David Harel: *Statecharts: A visual formalism for computer systems.* Science of Computer Programming 8, No. 3, pp. 231–274 (1987)

[Jensen 92] K. Jensen: *Coloured Petri Nets*, Vol. 1 of: EATCS Monographs on Theoretical Computer Science. Springer-Verlag (1992)

[Kindler 95] Ekkart Kindler: *Modularer Entwurf verteilter Systeme mit Petrinetzen.* PhD thesis, Technische Unviversität München (1995)

[Kindler, Reisig, Völzer, Walter 97] E. Kindler, W. Reisig, H. Völzer, R. Walter: *Petri net based verification of distributed algorithms: An example.* Formal Aspects of Computing 9, pp. 409–424 (1997)

[Kindler, Walter 95] E. Kindler, R. Walter: *Message passing mutex.* In: J. Desel (ed.): Structures in Concurrency Theory, Workshops in Computing, pp. 205–219 (May 1995) Springer-Verlag

[Lamport 86] Leslie Lamport: *The Mutual Exclusion Problem: Part I - A Theory of Interprocess Communication.* Journal of the ACM, Vol. 33, No. 2, pp. 313–326 (1986)

[Lynch 96] N. A. Lynch: *Distributed Algorithms.* Morgan Kaufmann Publishers, San Francisco, Calif. (1996)

[Mattern 89] Friedemann Mattern: *Verteilte Basisalgorithmen.* Informatik-Fachberichte 226, Springer-Verlag (1989)

[Milner 89] Robin Milner: *Communication and Concurrency.* International Series in Computer Science. Prentice Hall, Englewood Cliffs, N. J. (1989)

[Misra 91] J. Misra: *Phase synchronisation.* Information Processing Letters 38, pp. 101–105 (1991)

[Manna, Pnueli 92] Zohar Manna, Amir Pnueli: *The Temporal Logic of Reactive and Concurrent Systems.* Springer-Verlag (1992)

[Manna, Pnueli 95] Zohar Manna, Amir Pnueli: *Temporal Verification of Reactive Systems.* Springer-Verlag, Berlin (1995)

[Naimi, Trehel, Arnold 96] M. Naimi, M. Trehel, A. Arnold: *A log(n) distributed mutual exclusion algorithm based on path reversal.* Journal of Parallel and Distributed Computing 34, No. 1, p. 13 (1996)

[Owicki, Lamport 82] Susan Owicki, Leslie Lamport: *Proving liveness properties of concurrent programs.* ACM Transact. on Programming Languages and Systems 4, pp. 455–495 (1982)

[Peterson 81] G. L. Peterson: *Myths about the mutual exclusion problem.* Information Processing Letters 12, No. 3, pp.115–116 (June 1981)

[Peng, Makki 96] W. Peng, K. Makki: *Petri nets and self-stabilization of communication protocols.* Informatica 20, pp. 113–123 (1996)

[Raynal 88] M. Raynal: *Distributed Algorithms and Protocols.* Wiley Series in parallel computing. J. Wiley and Sons (1988)

[Raymond 89] K. Raymond: *A tree-based algorithm for distributed mutual exclusion.* ACM Transactions on Computer Systems 7, No. 1, pp. 61–77 (February 1989)

[Reisig 85] Wolfgang Reisig: *Petri nets.* Vol. 4 of: EATCS Monographs on Theoretical Computer Science. Springer-Verlag, Berlin (1985)

[Reisig 95] Wolfgang Reisig: *Petri net models of distributed algorithms.* In: Jan van Leeuwen (ed.): Computer Science Today. Recent Trends and Developments, Vol. 1000 of: LNCS Lecture Notes on Computer Science, pp. 441–454. Springer-Verlag, Berlin (1995)

[Reisig 96a] W. Reisig: *Interleaved progress, concurrent progress, and local progress.* In: D. A. Peled, V. R. Pratt, G. J. Holzmann (eds.): Partial Order Methods in Verification, Vol. 29, pp. 24–26 DIMACS Series in Discrete Mathematics and Theoretical Computer Science, American Mathematical Society (1996)

[Reisig 96b] W. Reisig: *Modeling and verification of distributed algorithms.* In: U. Montanari, V. Sassone (eds.): CONCUR 96: Concurrency Theory, vol. 1119 of: LNCS Lecture Notes in Computer Science, pp. 79–95 Springer-Verlag (1996)

[Reisig 98] W. Reisig: *Elements of Distributed Algorithms.* Springer-Verlag (1998)

[Raynal, Helary 90] M. Raynal, J. -M. Helary: *Synchronization and Control of Distributed Systems and Programs.* Wiley Series in parallel computing. J. Wiley and Sons (1990)

[Reisig, Kindler 97] W. Reisig, E. Kindler: *Verification of distributed algorithms with algebraic Petri nets.* In: C. Freksa, M. Jantzen, R. Valk (eds.): Foundations of Computer Science. Potential, Theory, Cognition, LNCS 1337, pp. 261–270 Springer-Verlag (1997)

[Rozenberg 86] G. Rozenberg: *Behaviour of elementary net systems.* In: W. Brauer, W. Reisig, G. Rozenberg (eds.): Petri Nets: Central Models and Their Properties, LNCS 254, pp. 60–94 Springer-Verlag (1986)

[Schneider 97] F. B. Schneider: *On Concurrent Programming.* Springer (1997)

[Segall 83] A. Segall: *Distributed network protocols.* IEEE Transactions on Information Theory, IT 29, No. 1, pp. 23–35 (1983)

[Shavit, Francez 86] N. Shavit, N. Francez: *A new approach to detection of locally indicative stability.* In: L. Kott (ed.): Proceedings of the 13th ICALP, LNCS 226, pp. 344–358 Springer-Verlag (1986)

[Tel 91] G. Tel: *Topics in Distributed Algorithms.* Cambridge International Series on Parallel Computation, 1, Cambridge University Press, Cambridge, U. K. (1991)

[Tel 94] G. Tel: *Introduction to Distributed Algorithms.* Cambridge University Press, Cambridge, U. K. (1994)

[Völzer 97] H. Völzer: *Verifying fault tolerance of distributed algorithms formally: An example.* In: 1998 International Conference on Application of Concurrency to System Design, Fukushima, Japan, (March 1998) IEEE Computer Society Press.

[Valk 86] Rüdiger Valk: *Infinite Behaviour and Fairness.* In: W. Brauer, W. Reisig, G. Rozenberg (eds.) Petri Nets: Central Models and Their Properties LNCS 254, pp. 377–396 Springer-Verlag(1986)

[Walter 95] R. Walter: *Petrinetzmodelle verteilter Algorithmen.* PhD thesis, Humboldt-Universität zu Berlin, Institut für Informatik. Edition Versal, vol.2 Bertz Verlag Berlin (1995)

[Walter 97] R. Walter: *The asynchronous stack revisited: Rounds set the twilight reeling.* In: C. Freksa, M. Jantzen, R. Valk (eds.) Foundations of Computer Science, LNCS 1337, pp. 307–312 Springer-Verlag (1997)

[Weber et al 98] M. Weber, R. Walter, H. Völzer, T. Vesper, W. Reisig, S. Peuker, E. Kindler, J. Freiheit, J. Desel: *DAWN: Petrinetzmodelle zur Verifikation verteilter Algorithmen.* Informatik-Bericht 88, Humboldt-Universität zu Berlin (1998)