

Algebraic System Nets for Modelling Distributed Algorithms*

Ekkart Kindler Wolfgang Reisig
Humboldt-Universität zu Berlin
Institut für Informatik
D-10099 Berlin

Introduction

Algebraic high-level nets [20, 3, 27, 21, 25, 1, 5, 13, 7, 23] have been introduced in order to exploit the rich theory of *algebraic specifications* [10, 2, 11] for high-level Petri nets. Besides the use of algebraic specifications most formalizations of algebraic nets have one feature in common: the number of tokens flowing through a particular arc when the corresponding transition occurs is always the same — in whichever mode the transition occurs. We say the arcs of this kind of algebraic nets have *fixed arc-weight* [26]. For modelling distributed algorithms this ‘feature’ turned out to be disadvantageous. This will be demonstrated by a simple example in Sect. 1. A restricted version of *flexible arc-weights* (*flexible arcs* for short) has already been introduced in P-Graphs [6, 5, 7] and been indicated as a possible extension for algebraic nets [23]. Here, we formalize a more general version of flexible arc, the idea of which can already been found in [7]. We call the resulting class *algebraic system nets*. The *P-invariant calculus* of Reisig [23] can be adapted to algebraic system nets, which will be demonstrated in a forthcoming paper [16]. We add two more features to algebraic system nets, which have been shown to be necessary for adequately modelling distributed systems [24]: the distinction of *progress* and *external* transitions and *fair arcs* [18].

When talking about a distributed algorithm we actually do not talk about a single algebraic system net, but a class of algebraic system nets. The reason is that distributed algorithms usually work for different communication networks, with varying numbers of nodes. In order to deal with a class of algebraic net systems we introduce *algebraic net schemes*, where the algebra over which a net is interpreted may vary. This aspect was one of the reasons for introducing algebraic nets [27].

The paper is organized as follows: In Sect. 1 we informally introduce algebraic system nets by two examples. These examples motivate the necessity of flexible arc-weights and

*This work was supported by the DFG (Projects ‘Distributed Algorithms’ and ‘Petri-net-Technology’).

of progress and fairness. Section 2 gives a self-contained formalization of algebraic system nets and their runs including all prerequisites from algebraic specifications and net theory. Nevertheless, we expect some basic knowledge of both theories.

1 Two Examples

Before introducing *algebraic system nets* formally we give two informal examples. The first example motivates the need for flexible arc-weights, the second motivates the need for the distinction of external and progress transitions and for fair arcs. These examples will be formalized in Sect. 2.

1.1 A minimum distance algorithm

The first algorithm works on a network of *agents* where some distinguished agents are so-called *roots* of the network. The algorithm computes for each agent of the network the minimal distance from a root. This algorithm was inspired by a simple spanning tree algorithm [8].

We denote the set of agents by A , the set of distinguished root-agents by $R \subseteq A$; the set of other so-called *inner* agents is denoted by $I = A \setminus R$. The underlying network is denoted by $N \subseteq A \times A$. The algebraic net system Σ_1 shown in Fig. 1 models the behaviour of each agent $x \in A$.

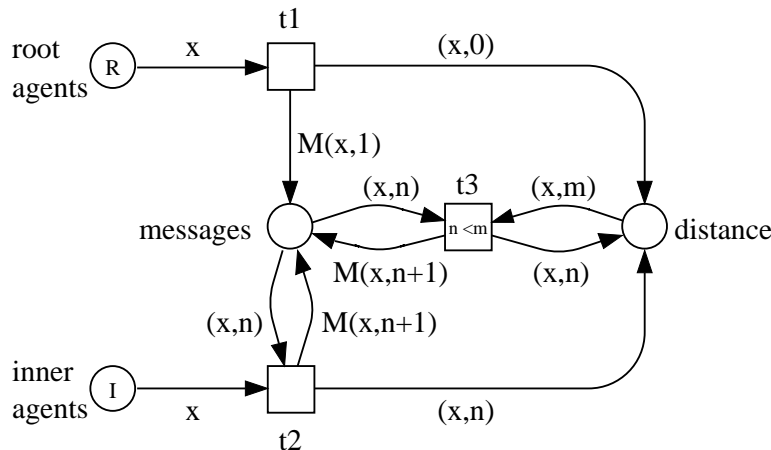


Figure 1: A minimum distance algorithm Σ_1

Initially, a root-agent $x \in R$ sends a message to each *neighbour* in the network. In this message he informs his neighbours that they have distance 1 from a root (viz. from x himself). The agent $x \in R$ makes an entry for himself that his distance from a root is 0. This behaviour is modelled by *transition* t_1 of Σ_1 : A message m to an agent $y \in A$ is represented as a pair (y, m) on place *messages*. This pair can be interpreted as a letter addressed to y containing the message m . If y_1, \dots, y_n are the neighbours of x in the communication network, $M(x, 1)$ denotes the set of pairs¹ $[(y_1, 1), \dots, (y_n, 1)]$ — one

¹The use of square brackets indicates that actually we use multisets, rather than sets.

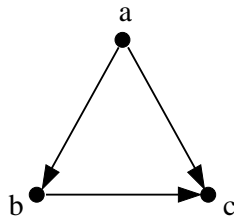


Figure 2: A network of three agents

letter for each neighbour.

An *inner agent* $x \in I$ waits until he receives a letter from one of his neighbours. When he receives a letter (x, n) he accepts the distance n ; in addition he sends a message $n + 1$ to each of his neighbours. This behaviour is modelled by transition t_2 .

When an agent $x \in A$ receives another letter (x, n) with a distance n which is shorter than the one he currently knows, he accepts the new distance n and sends the new distance $n + 1$ to each of his neighbours. This behaviour is modelled by transition t_3 , where the transition inscription $n < m$ guarantees that the new distance is shorter than the old one. Altogether, this behaviour guarantees that eventually each agent knows his minimal distance to a root — if there is a path to some root at all. The minimal distance n of an agent $x \in A$ is represented as a pair (x, n) on place *distance*.

Let us consider how messages are sent out in Σ_1 in more detail: As we said above, sending a message m to an agent x is modelled as a pair (x, m) on place *messages* — in our example m is a number representing distance. In order to get a simple and concise Petri net model of the algorithm we have modelled the sending of a message to all neighbours by a single transition; this is possible because $M(x, 1)$ resp. $M(x, n)$ represents a set of letters — one for each neighbour. Of course, the set denoted by $M(x, n)$ depends on the agent x and the underlying network N . For the network shown in Fig. 2 we have: $M(a, n) = [(b, n), (c, n)]$, $M(b, n) = [(c, n)]$, and $M(c, n) = []$ for each $n \in \mathbb{N}$. For this network the number of pairs (letters) in $M(x, n)$ varies for the different agents. Therefore, the number of tokens ‘flowing through’ the arc from t_1 to *messages* varies between 0 and 2. This is a typical example for a flexible arc. The flexible arcs of Σ_1 cannot be represented in many formalizations of algebraic nets (e.g. [23]).

Of course, it is possible to model the above algorithm by a conventional algebraic net. For example, one could send the messages to each neighbour one after the other. But, the resulting algebraic net has more transitions and is more complicated than Σ_1 ; the simplicity of Σ_1 results from the use of flexible arcs. Moreover, sending the messages to each neighbour in some fixed order is a design decision, which is completely irrelevant for the correctness of the algorithm. In this sense the above model represents the algorithmic idea more concisely.

Since sending messages is a basic feature of many distributed algorithms (cf. [28]) we should be able to represent it immediately in an algebraic system net.

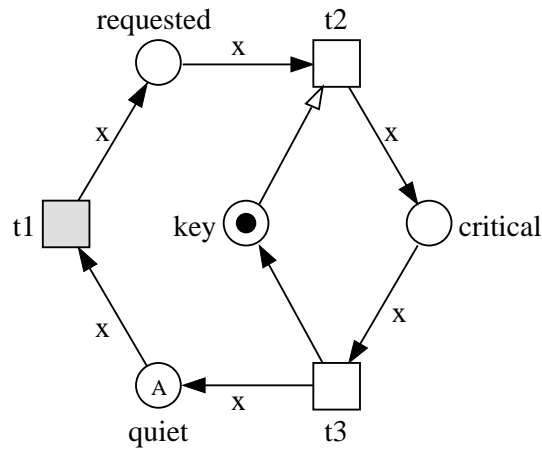


Figure 3: A simple algorithm for mutual exclusion: Σ_2

1.2 Mutual exclusion

Let us consider another example, which motivates two other features of algebraic system nets. Again, we have a set of *agents* A . The algorithm² guarantees that two agents are never in their so-called *critical section* at the same time (*mutex*). For simplicity we present the well-known semaphore solution.

The algorithm is modelled by the algebraic system net Σ_2 shown in Fig. 3. Initially, no agent is in his critical section. When an agent $x \in A$ wants to enter his critical section, he requests access (transition t_1). He may enter the critical section, when he can get hold of the key (transition t_2); since there is only one key (represented as a conventional black token) mutual exclusion is guaranteed. Eventually, the agent leaves the critical section and returns the key (transition t_3).

The decision of an agent to enter the critical section is in some sense up to a not modelled environment. Therefore, we call transition t_1 *external*, which is graphically represented as a shaded square. An external transition need not occur in a run — even when continuously enabled. For the other transitions we assume that they must eventually occur, when continuously enabled. Therefore, we call the other transitions *progress* transitions. For example an agent may not stay in his critical section forever; the progress transition t_3 must eventually occur. By help of external transitions we can formalize Dijkstra's [9] requirement, that a mutex-solution should work even when some agents will stop to request access to the critical section (cf. [19]).

Another concept incorporated in algebraic system nets is fairness. Consider the situation where an agent a enters the critical section over and over again and agent b requests access to the critical section; but, agent a always snatches away the key such that b can never get hold of it. Clearly, this is unfair with respect to agent b . We want to exclude such unfair assignment of the key. This fairness requirement is indicated by a *fair arc* from place *key* to transition t_2 , which is graphically represented by a white arrow-head. Technically, the fair arc has the following effect: When there is a token a on place *requested* and the token \bullet always 'returns' to place *key*, then transition t_2 will eventually occur in mode $x = a$.

²We do not claim that the presented algorithm is really distributed.

The effect of progress transitions and fair arcs on runs of an algebraic system net will be formalized in more detail later on. For now, it is sufficient to know that we need to distinguish external and progress transitions and that we need fair arcs.

2 Algebraic system nets

In this section we formalize algebraic system nets.

2.1 Basic notations

First, we introduce some notations and basic concepts from algebraic specifications [10] and Petri nets [22]. The only new concept is the *multiset-signature* along with a corresponding concept of a *multiset-algebra*: In a multiset-signature we distinguish some *ground-sorts* and we assign a *multiset-sort* to each ground-sort. In a multiset-algebra the domain associated with a multiset-sort must be a multiset over the domain of the corresponding ground-sort.

Sets, families, and mappings By \emptyset , \mathbb{N} , and \mathbb{B} we denote the *empty* set, the set of natural numbers with 0, and the set of truth values true and false, respectively. For a set A we write $a \in A$, if a is an element of A and $a \notin A$ otherwise. By $|A|$ we denote the cardinality of A , where ∞ stands for any infinite cardinality. The union and the intersection of two sets A_1 and A_2 is denoted by $A_1 \cup A_2$ and $A_1 \cap A_2$, respectively.

For a set A we denote the set of all non-empty finite sequences over A by A^+ . For some set I we call $(A_i)_{i \in I}$ a *family* of sets, if A_i is a set for each $i \in I$. The family $(A_i)_{i \in I}$ is *pairwise disjoint*, if for each $i, j \in I$ with $i \neq j$, $A_i \cap A_j = \emptyset$. For a family $A = (A_i)_{i \in I}$ we often use A to denote the union $\bigcup_{i \in I} A_i$.

For sets A_1, \dots, A_n we denote the *product* of these sets by $A_1 \times \dots \times A_n$. By $f : A_1 \times \dots \times A_n \rightarrow A_{n+1}$ we denote a total mapping from $A_1 \times \dots \times A_n$ to A_{n+1} .

Multisets In contrast to sets, an element can occur multiple times in a *multiset*. A multiset over a fixed set A is formalized as a mapping $m : A \rightarrow \mathbb{N}$. The set of all multisets over A is denoted by \mathbb{N}^A . In order to indicate syntactically that m is a multiset we write $m[a]$ instead of $m(a)$. Then, $m[a]$ denotes the number of occurrences of $a \in A$ in the multiset m .

We represent a finite multiset by enumerating its elements in square brackets: $[a_1, \dots, a_n]$. In particular, $[]$ denotes the empty multiset, i.e. $[] [a] = 0$ for each $a \in A$.

Addition $m_1 + m_2$ of two multisets m_1 and m_2 is defined element-wise by $(m_1 + m_2)[a] = m_1[a] + m_2[a]$ for each $a \in A$. Similarly, we define the relation \leq on multisets by $m_1 \leq m_2$, iff $m_1[a] \leq m_2[a]$ for each $a \in A$.

Sometimes it is convenient to allow infinite multiplicity of an element in a multiset. Formally, this can be represented as mapping $m : A \rightarrow \mathbb{N} \cup \{\infty\}$; the set of all *multisets with infinite multiplicity* over A is denoted by \mathbb{N}_∞^A .

For convenience we will adopt the view³ that for two sets $A \subseteq B$ each multiset m over A is also a multiset over B with $m[b] = 0$ for $b \in B \setminus A$. Similarly, we have $\mathbb{N}^A \subseteq \mathbb{N}_\infty^A$.

Algebras and signatures A (sorted) *algebra* consists of a family of sets and a set of operations on these sets. The operations of an algebra and their arity are called *signature*. Formally, a *signature* $SIG = (S, OP)$ consists of a finite set S of *sort symbols* and a pairwise disjoint family $OP = (OP_a)_{a \in S^+}$ of *operation symbols*. An operation symbol $o \in OP_{s_1 \dots s_n s_{n+1}}$ stands for an operation from sorts s_1, \dots, s_n to sort s_{n+1} . In particular, $o \in OP_s$ denotes a *constant* of sort $s \in S$.

For a signature $SIG = (S, OP)$ a *SIG-algebra* $\mathcal{A} = ((A_s)_{s \in S}, (f_o)_{o \in OP})$ consists of a family $A = (A_s)_{s \in S}$ of sets corresponding to the sort symbols and an operation f_o for each operation symbol $o \in OP$. For $o \in OP_{s_1 \dots s_n s_{n+1}}$ the operation f_o is a mapping $f_o : A_{s_1} \times \dots \times A_{s_n} \rightarrow A_{s_{n+1}}$. Each set A_s is called a *domain* of the algebra.

In the following we always assume that a signature SIG has a sort symbol $bool \in S$ and in each SIG -algebra the corresponding domain is $A_{bool} = \mathbb{B}$.

Variables and terms For a signature $SIG = (S, OP)$ we call a pairwise disjoint family $X = (X_s)_{s \in S}$ with $X \cap OP = \emptyset$ *SIG-variables* (or *variables* when SIG is clear from the context). From the variables and the operation symbols of the signature we can build *terms*. Each term is associated with a particular sort. The *set of SIG-terms with variables* X and sort s is denoted by $\mathbf{T}_s^{SIG}(X)$ and inductively defined by:

1. $X_s \subseteq \mathbf{T}_s^{SIG}(X)$.
2. For an operation symbol $o \in OP_{s_1 \dots s_n s_{n+1}}$ and terms $u_i \in \mathbf{T}_{s_i}^{SIG}(X)$ for $i = 1, \dots, n$ we have $o(u_1, \dots, u_n) \in \mathbf{T}_{s_{n+1}}^{SIG}(X)$.

The set of all terms (of any sort) is denoted by $\mathbf{T}^{SIG}(X)$. A term without variables is called a *ground term*. We denote the set of ground terms by $\mathbf{T}^{SIG} = \mathbf{T}^{SIG}(\emptyset)$ and the set of ground terms of sort s by $\mathbf{T}_s^{SIG} = \mathbf{T}_s^{SIG}(\emptyset)$.

Assignment and evaluation For a signature $SIG = (S, OP)$, SIG -variables $X = (X_s)_{s \in S}$, and a SIG -algebra $\mathcal{A} = ((A_s)_{s \in S}, (f_o)_{o \in OP})$ a mapping $\beta : X \rightarrow A$ is an *assignment*, if for each $s \in S$ and $x \in X_s$ holds $\beta(x) \in A_s$. An assignment associates each variable with a value of the corresponding domain. For a given assignment we can evaluate a term to a value; this is formalized as a mapping $\bar{\beta} : \mathbf{T}^{SIG}(X) \rightarrow A$ which is defined inductively over the structure of terms:

1. $\bar{\beta}(x) = \beta(x)$ for $x \in X$.
2. $\bar{\beta}(o(u_1, \dots, u_n)) = f_o(\bar{\beta}(u_1), \dots, \bar{\beta}(u_n))$ for $o(u_1, \dots, u_n) \in \mathbf{T}^{SIG}(X)$.

Mapping $\bar{\beta}$ is called the *extension of β* . For an empty variable set there exists exactly one mapping $\beta : \emptyset \rightarrow A$. The extension $\bar{\beta}$ of this assignment is called *evaluation* since it evaluates ground terms to values of the algebra. We will denote the evaluation by *eval* when SIG and \mathcal{A} are clear from the context.

³This view depends on a particular set-theoretic formalization of mappings, which is beyond the scope of this paper.

Multiset-signatures and -algebras In an algebraic system net a token on a place is an element of a domain of some algebra. Then, the multitude of tokens on a particular place can be represented as a multiset. In order to talk about the multitude of tokens on a place within the algebraic framework, we include the multiset over the corresponding sort in the algebra.

To this end, we introduce multiset-signatures as particular signatures. For a signature $SIG = (S, OP)$ and a subset $S' \subseteq S$ an injective mapping $ms : S' \rightarrow S$ associates each *ground-sort* $s \in S'$ with a corresponding *multiset-sort* $ms(s)$. We call $MSIG = (S, OP, ms)$ a *multiset-signature* with ground sorts S' . A SIG -algebra \mathcal{A} is a $MSIG$ -algebra, if for each ground-sort $s \in S'$, $A_{ms(s)} = \mathbb{N}^{A_s}$, i.e. if for each *ground-domain* the corresponding *multiset-domain* is actually a multiset over the ground-domain.

Note, that a multiset-signature $MSIG = (S, OP, ms)$ is a specialized signature $SIG = (S, OP)$ and by definition each $MSIG$ -algebra is a SIG -algebra. Therefore, terms, assignments, and the evaluation are well-defined for multiset-signatures, too.

Example 1

Here we present the multiset-signatures and multiset-algebras corresponding to the system nets Σ_1 and Σ_2 from Fig. 1 and 3.

Σ_1 : For the system net Σ_1 we implicitly used a multiset-algebra with the following multiset-signature: $MSIG^1 = (S^1, OP^1, ms^1)$ with sorts $S^1 = \{agent, nat, pair, msagent, mspair\}$, ground-sorts $S^1 = \{agent, pair\}$ and $ms^1 : S^1 \rightarrow S^1$ with $ms^1(agent) = msagent$ and $ms^1(pair) = mspair$. We used the operation symbols $(.,.) \in OP^1_{agent\ pair\ pair}$, $. + . \in OP^1_{nat\ nat\ nat}$, and $M \in OP^1_{agent\ nat\ mspair}$ and the constant symbols $0 \in OP^1_{nat}$, $1 \in OP^1_{nat}$, $R \in OP^1_{msagent}$, and $I \in OP^1_{msagents}$. Note, that we use mixfix- and infix-notation for the operation symbol $(.,.)$ and $. + .$, respectively, in order to get more readable terms.

For reasons which will become clear later on, we need two additional operation symbols $[.]_{agent} \in OP^1_{agent\ msagent}$ and $[.]_{pair} \in OP^1_{pair\ mspair}$.

The $MSIG^1$ -algebra \mathcal{A}^1 representing the network from Fig. 2 consists of the sets $A^1_{agent} = \{a, b, c\}$, $A^1_{nat} = \mathbb{N}$ and $A^1_{pair} = A^1_{agent} \times A^1_{nat}$, $A^1_{msagent} = \mathbb{N}^{A^1_{agent}}$, and $A^1_{mspair} = \mathbb{N}^{A^1_{pair}}$. The operation $f_{(.,.)} : A^1_{agent} \times A^1_{nat} \rightarrow A^1_{pair}$ is defined as expected $f_{(.,.)}(x, n) = (x, n)$ for each $x \in A^1_{agent}$ and $n \in A^1_{nat}$. The operation $f_{.+} : A^1_{nat} \times A^1_{nat} \rightarrow A^1_{nat}$ is the usual addition on natural numbers. The operation $f_M : A^1_{agent} \times A^1_{nat} \rightarrow A^1_{mspair}$ is defined as already indicated in Sect. 1.1: $f_M(a, n) = [(b, n), (c, n)]$, $f_M(b, n) = [(c, n)]$, and $f_M(a, n) = []$ for each $n \in \mathbb{N}$. We choose the constants $f_0 = 0$, $f_1 = 1$, $f_R = [a]$, and $f_I = [b, c]$.

For each ground-sort s the additional operation $f_{[.]_s}$ is defined by $f_{[.]_s}(x) = [x]$ for each $x \in A_s$.

In Σ_1 we use the following variables: $x \in X_{agent}$, and $n, m \in X_{nat}$.

Σ_2 : For the system net Σ_2 we used the following multiset-signature $MSIG^2 = (S^2, OP^2, ms^2)$ with sorts $S^2 = \{agent, dot, msagent, msdot\}$, ground-sorts

$S'^2 = \{agent, dot\}$, and $ms^2(agent) = msagent$ and $ms^2(dot) = msdot$. As operation symbols we have only constant symbols $A \in OP^2_{msagent}$ and $\bullet \in OP^2_{dot}$.

Again we introduce two additional operation symbols $[\cdot]_{agent} \in OP^2_{agent msagent}$ and $[\cdot]_{pair} \in OP^2_{dot msdot}$.

The $MSIG^2$ -algebra \mathcal{A}^2 consists of the sets $A^2_{agent} = \{a, b, c\}$, $A^2_{dot} = \{\bullet\}$, $A^2_{msagent} = \mathbb{N}^{A^2_{agent}}$, and $A^2_{msdot} = \mathbb{N}^{A^2_{dot}}$. The constants are chosen as follows $f_A = [a, b, c]$ and $f_\bullet = \bullet$.

Again for each ground-sort s the additional operation $f_{[\cdot]_s}$ is defined by $f_{[\cdot]_s}(x) = [x]$ for each $x \in A_s$.

In Σ_2 we use only one variable $x \in X_{agent}$.

Petri nets At last, we introduce *Petri nets*. A *net* $N = (P, T, F)$ consists of two disjoint sets P and T and $F \subseteq (P \times T) \cup (T \times P)$. An element of P is called *place*, an element of T is called *transition*, and an element of F is called *arc* of the net. As usual, we graphically represent the places by circles, the transitions by squares, and the arcs by arrows between the corresponding elements.

2.2 Algebraic system nets

Now, we are prepared to define *algebraic system nets*.

Definition 1 (Algebraic system nets)

Let $MSIG = (S, OP, ms)$ be a multiset-signature with ground-sorts S' . An *algebraic system net* Σ over $MSIG$ consists of

1. a net $N = (P, T, F)$,
2. a distinguished set $T_e \subseteq T$ of *external Transitions*,
3. a distinguished set $F_f \subseteq F \cap (P \times (T \setminus T_e))$ of *fair arcs*,
4. an $MSIG$ -Algebra \mathcal{A} ,
5. a set of $MSIG$ -variables X ,
6. a mapping $d : P \rightarrow S$,
7. a mapping $w : F \rightarrow \mathbf{T}^{MSIG}(X)$, such that for each $p \in P$ and $t \in T$ holds:
 $(p, t) \in F$ implies $w(p, t) \in \mathbf{T}^{MSIG}_{ms(d(p))}(X)$ and $(t, p) \in F$ implies $w(t, p) \in \mathbf{T}^{MSIG}_{ms(d(p))}(X)$
8. a mapping $g : T \rightarrow \mathbf{T}^{MSIG}_{bool}(X)$, and
9. a mapping $m_0 : P \rightarrow \mathbf{T}^{MSIG}$ such that $m_0(p) \in \mathbf{T}^{MSIG}_{ms(d(p))}$ for each $p \in P$.

We denote this algebraic system net by $\Sigma = (N, T_e, F_f, \mathcal{A}, X, d, w, g, m_0)$. We call N the *underlying net*, $T \setminus T_e$ the *progress transitions*, \mathcal{A} the *underlying MSIG-algebra*, d the *sort-function*, w the *arc-inscription*, g the *guard-function*, and m_0 the *symbolic initial marking* of Σ .

Note, that according to this definition neither Σ_1 nor Σ_2 along with the corresponding algebra from Example 1 is an algebraic system net! There are inscriptions that are no terms of a multiset-sort, but a term of a ground-sort: e.g. $x \in \mathbf{T}_{agents}^{MSIG}(X)$ and $(x, n) \in \mathbf{T}_{pairs}^{MSIG}(X)$. Since we frequently need inscriptions which correspond to singleton multisets, we introduce the following convention: for each ground-sort $s \in S$ there is an operation symbol $[\cdot]_s : s \rightarrow ms(s)$ (cf. additional operations in our example). In a MSIG-algebra the corresponding operation $f_{[\cdot]_s}$ maps each element of the ground-domain to the corresponding singleton multiset. Now, if we have an inscription $u \notin \mathbf{T}_{ms(d(p))}^{MSIG}(X)$, we actually mean $[u]_{d(p)}$. Moreover, we assume that each transition, which has no inscription in its graphical representation is inscribed by true and an unscribed arc is inscribed by $[\bullet]$. By this convention both, Σ_1 and Σ_2 with the corresponding algebras \mathcal{A}^1 and \mathcal{A}^2 from Example 1, are algebraic system nets.

In a sense Definition 1 gives the syntax of algebraic systems net. Still, the algebra is given semantically because we want to be flexible. We can incorporate any appropriate algebraic formalism for representing an algebra. The semantics, i.e. the runs of an algebraic system net, will be defined in Sect. 2.3. Here, we only define *markings* and the *firing-rule* for algebraic system nets. A marking associates with each place of an algebraic system net a multiset of the corresponding sort.

Definition 2 (Marking and initial marking)

Let MSIG be a multiset-signature and Σ be an algebraic system net as in Def. 1. A mapping $M : P \rightarrow \mathbb{N}^A$ is a *marking* of Σ , if for each place $p \in P$ holds $M(p) \in \mathbb{N}^{A_{d(p)}}$. The marking M_0 with $M_0(p) = eval(m_0(p))$ for each $p \in P$ is called the *initial marking* of Σ .

We define the *addition* $M_1 + M_2$ of two markings place-wise: $(M_1 + M_2)(p) = M_1(p) + M_2(p)$. The relation \leq on markings is defined by: $M_1 \leq M_2$, iff for each $p \in P$ holds $M_1(p) \leq M_2(p)$.

A *mode* of a transition associates a value of a domain of the algebra with each variable of X ; technically a firing-mode is just an assignment for variables X . In a particular mode, an arc-inscription evaluates to some multiset. A transition t may fire in mode β , if the guard $g(t)$ evaluates to true for β and all elements denoted by the evaluation of the arc-inscription of each arc from p to t , are present at place p in the current marking.

We formalize the firing-rule by associating each transition t and each mode β with a marking $^-t_\beta$ and a marking $^+t_\beta$. The marking $^-t_\beta$ represents the elements which are removed from the corresponding place when t fires in mode β , the marking $^+t_\beta$ represents the elements which are added to the corresponding place when t fires in mode β .

Definition 3

Let $MSIG$ be a multiset-signature and Σ be an algebraic system net as in Def. 1. Let $t \in T$ and β be an assignment of X in \mathcal{A} . We define the two markings ${}^{-}t_{\beta} : P \rightarrow \mathbb{N}^A$ and $t_{\beta}^{+} : P \rightarrow \mathbb{N}^A$, such that for each $p \in P$

$${}^{-}t_{\beta}(p) = \begin{cases} \overline{\beta}(w(p, t)) & \text{for } (p, t) \in F \\ [] & \text{for } (p, t) \notin F \end{cases}$$

and

$$t_{\beta}^{+}(p) = \begin{cases} \overline{\beta}(w(t, p)) & \text{for } (t, p) \in F \\ [] & \text{for } (p, t) \notin F \end{cases}$$

In a given marking M a transition t is *enabled* in mode β , if $\overline{\beta}(g(t)) = \text{true}$ and ${}^{-}t_{\beta} \leq M$. Then, transition t can fire in mode β , which results in a *successor marking* M' , which is uniquely characterized by $M' + {}^{-}t_{\beta} = M + t_{\beta}^{+}$. Usually, the occurrence of transition t in mode β is denoted by $M \xrightarrow{t, \beta} M'$.

In the following we will only consider algebraic system nets⁴ for which ${}^{-}t_{\beta}$ and t_{β}^{+} are nonempty markings for each transition t and each mode β . This helps to avoid some anomalies in the definition of runs (see [4] for further explanation).

2.3 Runs of algebraic system nets

In this paper we are not only interested in the firing-rule, but in processes and runs of algebraic system nets since the concepts of *progress* and *fairness* are only sensible in these terms. We use non-sequential runs [12, 4] since they are particularly suited for this purpose.

Figure 4 shows an example of a run of the algebraic system net Σ_2 from Fig. 3 with algebra \mathcal{A}^2 : it is a particular net with place inscriptions corresponding to the algebraic system net. For a place $p \in P$ of the algebraic system net and an element $a \in A_{d(p)}$ the inscription (p, a) represents one token a on place p .

Basic concepts Before we can formalize runs we need some prerequisites, which mainly follow the lines of [4].

Definition 4

Let $N = (P, T, F)$ be a net.

1. For an element $x \in P \cup T$ of N we define the *preset* of x by $\bullet x = \{y \in P \cup T \mid (y, x) \in F\}$ and the *postset* of x by $x \bullet = \{y \in P \cup T \mid (x, y) \in F\}$.
2. We define the *minimal elements* of N by ${}^{\circ}N = \{x \in P \cup T \mid \bullet x = \emptyset\}$ and the *maximal elements* of N by $N^{\circ} = \{x \in P \cup T \mid x \bullet = \emptyset\}$.

⁴Maybe, we will restrict the definition of algebraic system nets to these systems in the future.

3. For $x \in P \cup T$ we define the *set of predecessors* by $\downarrow x = \{y \in P \cup T \mid (y, x) \in F^+\}$, where F^+ denotes the transitive closure⁵ of the flow relation F .

The nets which are used in the definition of runs are called *occurrence nets*. An *occurrence net* has two main features: the flow relation is acyclic and is not branching at places. Moreover, each element of an occurrence net has only finitely many predecessors. For a detailed motivation of all features we refer to [12, 4].

Definition 5 (Occurrence net)

A net $K = (B, E, \triangleleft)$ is an *occurrence net*, if

1. ${}^\circ K \subseteq B$ and $K^\circ \subseteq B$,
2. for each $b \in B$ holds $|\bullet b| \leq 1$ and $|b^\bullet| \leq 1$, and
3. for each $b \in B$ the set of predecessors $\downarrow b$ is finite and $b \notin \downarrow b$.

For clearness, we use new symbols for places and transitions of an occurrence net. Moreover, we call a place of an occurrence net a *condition* and a transition an *event*. Next we define the *states* of an occurrence net.

Definition 6 (States of an occurrence net)

Let $K = (B, E, \triangleleft)$ be an occurrence net. For subsets of conditions $Q, Q' \subseteq B$ we define the occurrence relation \rightarrow by: $Q \rightarrow Q'$ iff there exists an event $e \in E$ such that $\bullet e \subseteq Q$ and $Q' = (Q \setminus \bullet e) \cup e^\bullet$. The transitive and reflexive closure of \rightarrow is denoted by $\xrightarrow{*}$. For $Q, Q' \subseteq B$ we say Q' is *reachable from* Q , if $Q \xrightarrow{*} Q'$.

A subset of conditions $Q \subseteq B$ is a *state of* K , if Q is reachable from ${}^\circ K$. The set ${}^\circ K$ is called the *initial state* of K .

Processes and runs of algebraic net systems In a run each condition of the occurrence net is associated with some place of the algebraic net system along with an element of the corresponding domain. This is formalized as Σ -inscription.

Definition 7 (Σ -inscription)

Let Σ be an algebraic system net over a multiset-signature *MSIG* as in Def. 1, and $K = (B, E, \triangleleft)$ be an occurrence net. A mapping $r : B \rightarrow P \times A$ is a Σ -*inscription of* K , if for each $b \in B$ with $r(b) = (p, a)$ holds $a \in A_{d(p)}$.

For a given Σ -inscription r each subset $Q \subseteq B$ can be associated with a marking (possibly with infinite multiplicity). We denote this marking by $r(Q)$ and define it by $r(Q) : P \rightarrow \mathbb{N}_\infty^A$ with $r(Q)(p)[a] = |\{b \in Q \mid r(b) = (p, a)\}|$.

Now, a Σ -inscribed occurrence net is a *process* of an algebraic system net Σ , if the initial state corresponds to the initial marking of Σ according to the inscription and each event corresponds to the firing of a transition of Σ in some mode.

⁵Note, that we do not use the transitive and reflexive closure of F ! This way, we can express acyclicity of F by $p \notin \downarrow p$ for each place $p \in P$

Definition 8 (Process of an algebraic system net)

Let Σ be an algebraic system net, $K = (B, E, \triangleleft)$ be an occurrence net and r be a Σ -inscription of K . Then (K, r) is a *process* of Σ , if

1. $r(\circ K) = M_0$, where M_0 is the initial marking of Σ , and
2. for each event $e \in E$ there exists a transition $t \in T$ and a mode β such that $\overline{\beta}(g(t)) = \text{true}$, $r(\bullet e) = \neg t_\beta$, and $r(e\bullet) = t_\beta^+$.

Definition 8 is the canonical extension of processes [4] to algebraic system nets. In addition to a process a *run* must satisfy *progress* and *fairness*, which were informally introduced in Sect. 1.2. Progress means that no enabled progress transition can be added at the end of the process, where we consider the set of conditions K° as the *end of the process*. Fairness is defined in a very similar way: suppose at the end of a process only element a on place p is missing to enable transition t in some mode β ; moreover, suppose a occurs infinitely many times on place p in the considered process, then the arc (p, t) is treated in an unfair way in this process. In a run all fair arcs must be treated in a fair way.

Definition 9 (Run of an algebraic net system)

Let Σ be an algebraic system net and (K, r) be a process of Σ as in Def. 8. The process (K, r) is a *run* of Σ , if

1. for each progress transition $t \in T \setminus T_e$, each mode β of Σ , and each $Q \subseteq K^\circ$ such that $r(Q) = \neg t_\beta$ holds $\overline{\beta}(g(t)) = \text{false}$ and
2. for each fair arc $(p, t) \in F_f$, each mode β of Σ , and each $Q \subseteq K^\circ$ such that there exist infinitely many $b \in B$ with $r(Q \cup \{b\}) = \neg t_\beta$ holds $\overline{\beta}(g(t)) = \text{false}$

Figure 4 shows an example of a run of the algebraic system net Σ_2 with algebra \mathcal{A}^2 .

2.4 Algebraic net schemes

An algebraic system net is equipped with a fixed algebra. One objective for introducing algebraic nets was to interpret a net over a class of algebras. For example, the minimal distance algorithm from Sect. 1 works for an arbitrary network of agents; we can vary the set of agents A as well as the underlying network. The structure of the network is reflected implicitly in the function M .

We call a net which can be interpreted over a class of algebras an *algebraic net scheme*. The definition of an algebraic net scheme is almost identical to the definition of an algebraic system net; the only difference is that we do not fix a particular *MSIG*-Algebra, but give a class of algebras. Note, that we do not fix a formalism for representing a class of *MSIG*-algebras. Again, the reason is that we want to be flexible to use any appropriate formalism.

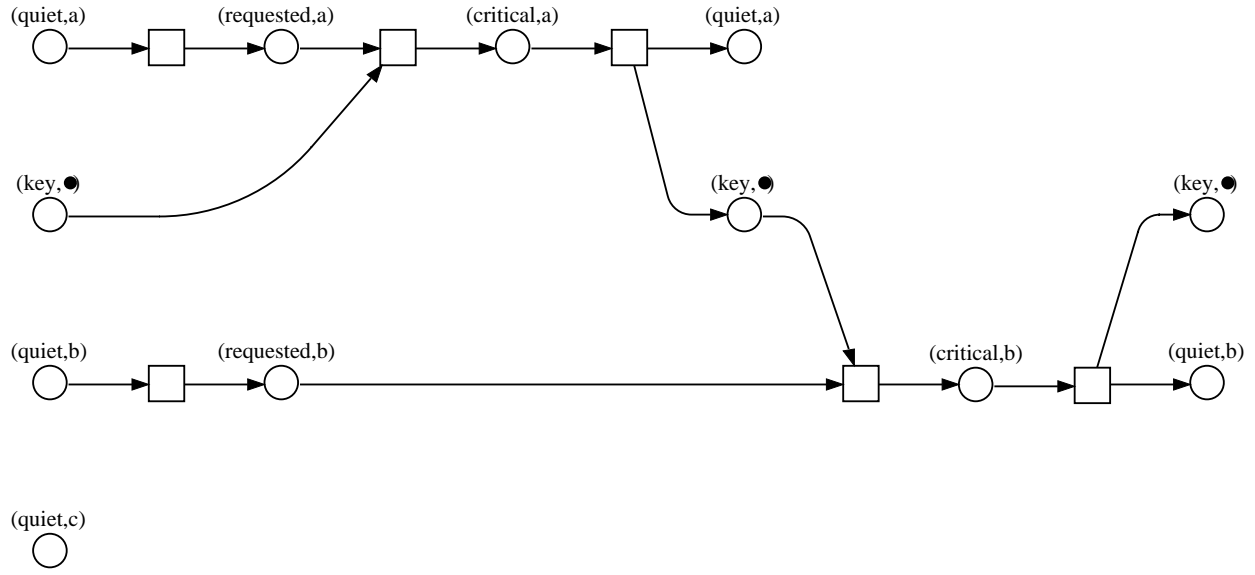


Figure 4: A run of Σ_2

Definition 10 (Algebraic net scheme)

Let $MSIG = (S, OP, ms)$ be a multiset-signature with ground-sorts S' . An *algebraic net scheme* Γ over $MSIG$ consists of

1. a net $N = (P, T, F)$,
2. a distinguished set $T_e \subseteq T$ of *external Transitions*,
3. a distinguished set $F_f \subseteq F \cap (P \times (T \setminus T_e))$ of *fair arcs*,
4. a class Λ of $MSIG$ -Algebras,
5. a set of $MSIG$ -variables X ,
6. a mapping $d : P \rightarrow S$,
7. a mapping $w : F \rightarrow \mathbf{T}^{MSIG}(X)$, such that for each $p \in P$ and $t \in T$ holds:
 $(p, t) \in F$ implies $w(p, t) \in \mathbf{T}_{ms(d(p))}^{MSIG}(X)$ and $(t, p) \in F$ implies $w(t, p) \in \mathbf{T}_{ms(d(p))}^{MSIG}(X)$
8. a mapping $g : T \rightarrow \mathbf{T}_{bool}^{MSIG}(X)$, and
9. a mapping $m_0 : P \rightarrow \mathbf{T}^{MSIG}$ such that $m_0(p) \in \mathbf{T}_{ms(d(p))}^{MSIG}$ for each $p \in P$.

We denote the algebraic net scheme by $\Gamma = (N, T_e, F_f, \Lambda, X, d, w, g, m_0)$.

The semantics of an algebraic net scheme is a class of algebraic system nets; an element of this class is called a model of Γ . The definition is straight forward.

Definition 11

Let $MSIG$ be multiset-signature and $\Gamma = (N, T_e, F_f, \Lambda, X, d, w, g, m_0)$ an algebraic net scheme over $MSIG$. Let $\mathcal{A} \in \Lambda$, then $\Sigma = (N, T_e, F_f, \mathcal{A}, X, d, w, g, m_0)$ is a *model* of Γ .

Note, that we do not define the concepts of a marking, a firing-rule, or a run for algebraic net schemes. The only purpose of an algebraic net scheme is to characterize a class of algebraic system nets — and to help to verify properties which this class has in common (cf. [28]).

3 Conclusion

We have defined algebraic system nets in a style similar to [23], but which allow flexible arc-weights. In a sense, an algebraic system net can be considered as a special representation of a *coloured net* [14, 15]. Yet a particularly nice one, which allows an almost seamless transition from modelling a single system to modelling a class of systems.

In addition to the firing-rule, we have defined the semantics of an algebraic system net in terms of their runs. This way we took care of *progress* and *fairness*, which are fundamental concepts for concisely modelling distributed algorithms. Our concept of algebraic nets was very much inspired by the work of Vautherin and Reisig [27, 23]; because we were heading for flexible arc-weights we could not keep up the strict separation between ground-sorts and multiset-sorts — we must allow ‘user-defined’ functions on multiset domains. Moreover, we do not insist on a particular kind of formalism for a syntactical representation of an algebra in order to achieve some flexibility and independence from a particular formalism from algebraic specifications. We have already proposed a similar formalism for algebraic nets [28] which is presented in a slightly more semantical way; e.g. the concept of signatures was only used implicitly. Here, we have chosen to use signatures explicitly in order to provide a clear interface to algebraic specifications.

Arguing about a class of systems rather than about one fixed system has been one motivation for algebraic nets from the first beginning [27, 6]. Our notion of algebraic net schemes seems to be particularly suited for representing distributed network algorithms (cf. [28]): essentially, each agent in the network performs the same algorithm; therefore, all agents can be folded to one net with the same structure. The structure of the communication-network is reflected in the algebra and not in structure of the underlying net. This allows to verify a distributed algorithm for arbitrary (or a particular kind of) communication networks. The presentation of these verification techniques is beyond the scope of this paper; they can be found in [17, 28].

The distinction of algebraic net schemes and algebraic system nets is fundamental. Nevertheless, it is only a slight technical difference (we replace ‘algebra’ by ‘class of algebras’ in the definition). In fact one can verify a distributed algorithm for a class of communication-networks without explicitly using algebraic net schemes in the following mathematical style:

We start ‘Let \mathcal{A} be a *MSIG*-Algebra with such and such properties.’ Then we do the verification by using the assumptions on the fixed algebra. We end with ‘Therefore, we have shown that the algorithm works for all algebras with such and such properties’.

This shows that one can deal with schemes on a mathematical level without an explicit definition. We have chosen to introduce both concepts explicitly for two reasons: First,

we wanted to elaborate on this fundamental difference. Second, we want to provide a clear interface to the theory of algebraic specifications on both levels, algebraic system nets and algebraic net schemes. In fact, algebraic specifications have better techniques to deal with classes of algebras than with a single algebra.

Acknowledgments The idea of this paper almost dates back five years! The formalization, its presentation, as well as our objective has changed over the years; different versions have been presented and discussed at internal meetings (in Munich and Berlin), working groups, workshops, and conferences. Consequently, many contributed to the development of our idea and the presentation of this paper, including J. Billington, J. Desel, H. Ehrig, D. Gomm, A. Heise, J. Lilius, J. Padberg, K. Schmidt, E. Smith, T. Thielke, T. Vesper, H. Völzer, and R. Walter. Thanks to all who contributed to this paper.

References

1. E. Battiston, F. De Cindio, and G. Mauri. OBJSA Nets: A class of highlevel nets having objects as domains. In G. Rozenberg, editor, *Advances in Petri Nets, LNCS 340*. Springer-Verlag, 1988.
2. J.A. Bergstra, J. Heering, and P. Klint (Eds.). *Algebraic Specification*. ACM Press. Addison Wesley, 1989.
3. B. Berthomieu, N. Choquet, C. Colin, B. Loyer, JM. Martin, and A. Mauboussin. Abstract Data Nets combining Petri nets and abstract data types for high level specification of distributed systems. In *Proceedings of VII European Workshop on Application and Theory of Petri Nets*, 1986.
4. Eike Best and César Fernández. *Nonsequential Processes, EATCS Monographs on Theoretical Computer Science 13*. Springer-Verlag, 1988.
5. J. Billington. Many-sorted high-level nets. In *Proceedings of the 3rd International Workshop on Petri Nets and Performance Models*, pages 166–179. IEEE Computer Society Press, December 1989.
6. Jonathan Billington. Extending coloured Petri nets. Technical Report 148, University of Cambridge, Computer Laboratory, October 1988.
7. Jonathan Billington. *Extensions to Coloured Petri Nets and their Application to Protocols*. Technical report no. 222, University of Cambridge, May 1991.
8. Manfred Broy. On the design and verification of a simple distributed spanning tree algorithm. SFB-Bericht 342/24/90 A, Technische Universität München, December 1990.
9. E. W. Dijkstra. Solution of a problem in concurrent programming control. *Communication of the ACM*, 8(9):569, 1965.
10. Hartmut Ehrig and Bernd Mahr. *Fundamentals of Algebraic Specifications 1, Equations and Initial Semantics, EATCS Monographs on Theoretical Computer Science 6*. Springer-Verlag, 1985.
11. Hartmut Ehrig and Bernd Mahr. *Fundamentals of Algebraic Specifications 2, Module Specifications and Constraints, EATCS Monographs on Theoretical Computer Sci-*

- ence 21. Springer-Verlag, 1990.
12. U. Goltz and W. Reisig. The non-sequential behaviour of Petri nets. *Information and Control*, 57:125–147, 1983.
 13. Udo Hummert. *Algebraische Theorie von high-level Netzen*. PhD thesis, Technische Universität Berlin, 1989.
 14. Kurt Jensen. *Coloured Petri Nets, Volume 1: Basic Concepts*. EATCS Monographs on Theoretical Computer Science. Springer-Verlag, 1992.
 15. Kurt Jensen. *Coloured Petri Nets. Volume 2: Analysis Methods*. EATCS Monographs in Theoretical Computer Science. Springer-Verlag, 1995.
 16. Ekkart Kindler, Wolfgang Reisig, and Hagen Völzer. Flexibility in algebraic nets. In preparation, 1996.
 17. Ekkart Kindler, Wolfgang Reisig, Hagen Völzer, and Rolf Walter. Petri net based verification of distributed algorithms: An example. Informatik-Berichte 63, Humboldt-Universität zu Berlin, May 1996.
 18. Ekkart Kindler and Rolf Walter. Message passing mutex. In J. Desel, editor, *Structures in Concurrency Theory*, Workshops in Computing, pages 205–219, Berlin, May 1995. Springer-Verlag.
 19. Ekkart Kindler and Rolf Walter. Mutex needs fairness. Submitted to a journal, August 1996.
 20. Bernd Krämer. Stepwise construction of non-sequential software systems using a net-based specification language. In G. Rozenberg with cooperation of H. Genrich and G. Roucairol, editors, *Advances in Petri Nets 1984*, LNCS 188, pages 307–330. Springer-Verlag, 1985.
 21. Bernd Krämer and Heinz-Wilhelm Schmidt. Types and modules for net specifications. In K. Voss, H.J. Genrich, and G. Rozenberg, editors, *Concurrency and Nets*. Springer-Verlag, 1987.
 22. Wolfgang Reisig. *Petri Nets, EATCS Monographs on Theoretical Computer Science 4*. Springer-Verlag, 1985.
 23. Wolfgang Reisig. Petri nets and algebraic specifications. *Theoretical Computer Science*, 80:1–34, May 1991.
 24. Wolfgang Reisig. Petri net models of distributed algorithms. In Jan van Leeuwen, editor, *Computer Science Today: Recent Trends and Developments*, LNCS 1000, pages 441–454. Springer-Verlag, 1995.
 25. Wolfgang Reisig and Jacques Vautherin. An algebraic approach to high level Petri nets. In *Proceedings of VIII European Workshop on Application and Theory of Petri Nets*, 1987.
 26. Einar Smith and Wolfgang Reisig. The semantics of a net is a net, an exercise in general net theory. In K. Voss, H.J. Genrich, and G. Rozenberg, editors, *Concurrency and Nets*. Springer-Verlag, 1987.
 27. Jacques Vautherin. Parallel systems specifications with coloured Petri nets and algebraic specifications. In G. Rozenberg, editor, *Advances in Petri Nets*, LNCS 266, pages 293–308. Springer-Verlag, 1987.
 28. R. Walter, H. Völzer, T. Vesper, W. Reisig, E. Kindler, J. Freiheit, and J. Desel. Memorandum: Petrinetzmodelle zur Verifikation verteilter Algorithmen. Informatik-Bericht 67, Humboldt-Universität zu Berlin, July 1996.