HELSINKI UNIVERSITY OF TECHNOLOGY
DIGITAL SYSTEMS LABORATORY
Series **A:** Research Reports ISSN 0783–5396
No. **33**; March 1995 ISBN 951–22–2522-0

ON THE STRUCTURE OF HIGH-LEVEL NETS

Johan Lilius

Digital Systems Laboratory
Department of Computer Science
Helsinki University of Technology
Otaniemi, FINLAND

Helsinki University of Technology
Department of Computer Science
Digital Systems Laboratory
Otaniemi, Otakaari 1
FIN–02150 ESPOO, FINLAND

HELSINKI UNIVERSITY OF TECHNOLOGY
DIGITAL SYSTEMS LABORATORY
Series **A**: Research Reports
No. **33**; March 1995

ISSN 0783–5396
ISBN 951–22–2522-0

On the Structure of High-level Nets

JOHAN LILIUS

Abstract: The structure of High-level nets is studied from an algebraic and a logical point of view using Algebraic nets as an example. First the category of Algebraic nets is defined and the semantics given through an unfolding construction. Other kinds of High-level net formalisms are then presented. It is shown that nets given in these formalisms can be transformed into equivalent Algebraic nets. Then the semantics of nets in terms of universal constructions is discussed. A definition of Algebraic nets in terms of structured transition systems is proposed. The semantics of the Algebraic net is then given as a free completion of this structured transition system to a category. As an alternative also a sheaf semantics of nets is examined. Here the semantics of the net arises as a limit of a diagram of sheaves. Next Algebraic nets are characterized as encodings of special morphisms called foldings. Each algebraic net gives rise to a surjective morphism between Petri nets and conversely each surjective morphism between Petri nets gives rise to an algebraic net. Finally it is shown how Algebraic nets can be described as sets of formulae in a typed version of intuitionistic predicate linear logic.

Keywords: net theory, category theory, algebraic specification, linear logic, petri nets, high-level nets.

Dissertation for the degree of Doctor of Technology to be presented with due permission for public examination and debate in Auditorium S4 at Helsinki University of Technology (Espoo, Finland) on the 16th of March, 1995, at 12 o'clock noon.

Printing: TKK Monistamo; Otaniemi 1995

Helsinki University of Technology
Department of Computer Science
Digital Systems Laboratory
Otaniemi, Otakaari 1
FIN–02150 ESPOO, FINLAND

Phone: $\frac{90}{+358\text{–}0}$ 4511
Telex: 125 161 htkk fi
Telefax: +358–0–465 077
E-mail: lab@saturn.hut.fi

Foreword

The work covered in this thesis has been carried out at the Digital Systems Laboratory of Helsinki University of Technology during the period 1990-1994. I am grateful to the head of the laboratory, prof. Leo Ojala, for providing excellent working conditions and for supporting me during this period.

I am greatly indebted to Dr. Nisse Husberg for his continuous support of this work.

I would also like to thank my other colleagues at the Digital Systems Laboratory for creating a stimulating working environment. Especially I would like to thank Dr. Ilkka Niemelä for very fruitful discussion during the last year.

In January 1991 I had the opportunity to visit Prof. Joseph Goguen and his group at Oxford University. As a result the work in section 3.2 was produced with support by the Information Technology Promotion Agency, Japan, as part of the R & D of Basic Technology for Future Industries "New Models of Software Architecture" project sponsored by NEDO (New Energy and Industrial Technology Developments Organization). I owe prof. Goguen my deepest gratitude.

I also wish to thank Julia Padberg, Leila Ribeiro and Prof. Hartmut Ehrig for constructive criticism on chapter 2.

I gratefully acknowledge the financial support received from the following foundations: Foundation for Financial Aid at Helsinki University of Technology, Foundation of Technology, Jenny and Antti Wihuri Foundation, and Oscar Öflund Foundation.

Finally, my biggest thanks go to my wife Kati for her support, encouragement and patience.

Espoo, October 1994

Johan Lilius

Contents

Chapter 1

Introduction

Within the field of information technology one of the fastest growing areas is the telecommunications industry. As we become more and more dependent on the connectivity offered by telecommunications networks, the correctness and dependability of these systems becomes an even more important issue. However, as the complexity of these systems increases it also becomes more and more difficult to ensure their correct functioning. Telecommunication systems are instances of parallel and distributed systems. The theory of parallel and distributed systems aims at providing tools for the analysis and modeling of such systems. One of the older approaches in this area is the theory of Petri nets (Petri 1962). Fundamental to Petri nets is the notion of causality, in that a Petri net describes the causal relations of events or actions in a system. Concurrency is then viewed as orthogonal to causality, meaning that if two events are not causally related then they can occur concurrently. Petri nets were soon recognized as an efficient notation for the description of parallel and distributed systems. It was however also soon recognized that Petri nets as such, formed an inadequate notation for the description of large systems. The problem is sometimes referred to as the "football field" problem, because the resulting Petri nets were huge. To overcome this growth in the size of the description, several so called high-level net formalisms have been proposed. They are all based on the recognition that often in a Petri net model, one has several subnets that are instances of the same process in the system, and that these subnets only differ in the parameters of the processes. High-level nets formalize this observation into the notion of an individual token. While in a Petri net each token is a "black dot" which contains no information whatsoever in itself, a token in a high-level net can contain information. The process for obtaining a high-level net from a low-level net is called folding. The basic idea is to fold the identical subnets, and to add so called annotations to components of the net, thus obtaining a more compact representation of the system.

In this thesis we will study a class of high-level nets called Algebraic nets. Algebraic nets use the theory of abstract datatypes as their annotation formalism, and aim at a discussion of the algebraic and logical foundations of these nets. On a more general level we try to give an answer to the general question of, to paraphrase Martin Heidegger, "What is it – a high-level net?". We will thus want to understand the basic mechanisms underlying the definition of high-level net classes. Using category theory we will try to give constructions that given certain basic concepts like a categorical formulation of Petri nets and the theory of substitutions, yield high-level nets. It should also be made clear, that we do not aim at explaining every possible feature exhibited in some form of high-level net, nor do we aim at giving the definite high-level net class that incorporates these features. As stated previously, our interest is in understanding the basic nature of high-level nets. To this end we will be working within a very simple high-level net class that actually incorporates very few features.

1.1 Modeling systems with high-level nets

In this section we will briefly look at the intuition behind high-level nets. Suppose we were given the task of modeling the following railway system. A circular railway consists of seven sections. Two trains operate the railway. The trains move freely in the same direction subject to the constraint, imposed for security reasons, that no two adjacent railway sections may be occupied at the same time. This is a classical example proposed by Genrich (1986). His C/E model is given in figure 1.1. The annotations of this net should be interpreted as follows. We have seven sections, $i = 0 \ldots 6$, and two trains $x = a, b$. Uix means that section i is occupied by train x, while Vi means that section i is vacant. In the initial marking train a is in section 0 and train b is in section 4, while sections $1, 2, 5$ are marked as vacant. Thus train a can move into section 1 and train b can move into section 5.

In this model there is a lot of conceptual duplication. If we think of trains and sections as individuals, we see that all the places Uix are instances of a single place U which is marked by a pair $\langle i, x\rangle$ that represents the fact that train x is in section i. Using the same idea we have a single place V which is marked with tokens $i = 1, 2, 5$ representing the vacant sections. If we take these two places as the starting point for the construction of our high-level net it turns out that we only need one transition as shown in figure 1.2. However, now the synchronization constraint that no two adjacent sections may be occupied at the same time, has to be expressed in the annotation formalism. This is taken care by the functions $f(i)$ and $g(i)$. Although it might seem that we have obtained the net in figure 1.2 from the one in figure 1.1 in an ad-hoc manner, there exists a strict mathematical relationship between the two nets: the net in figure 1.1 is the *unfolding* of the net in figure 1.2. Using this exact relationship we can also define the dynamics of the high-level net in terms of the low-level net. Fundamental to this semantics is the idea of substitution. For each variable of a transition we can substitute a value from its domain, ie. in figure 1.2 a valid substitution for t would be $\{i \mapsto 2, x \mapsto a\}$. If we instantiate the transition and evaluate its arc-expressions, we obtain the transition (given in a suggestive notation) $U(\langle 2, a\rangle) + V(3) \to U(\langle 3, a\rangle) + V(1)$. It is now easy to see, that this instance of t corresponds to the transition in figure 1.1 that is connected to the places $U2a, V3, U3a, V1$. Allowing substitutions on places gives us a notion of high-level marking: a substitution $\{U \mapsto \langle 2, a\rangle\}$ is interpreted as U is marked with token $\langle 2, a\rangle$. The firing rule of Petri nets can now also be lifted to the level of high-level nets, by the use of transition instances as expressed above. If there exists a substitution such that the terms in the input-places of a transition unify with the corresponding arc-annotations the transition may fire. It is important to note that the substitution only applies to one transition and thus the unification process is local to a transition. We thus have three fundamental constructions that concern high-level nets:

1. the semantics through unfolding,
2. the high-level semantics, and
3. the folding construction.

These will be the basic problems that we will study in this thesis.

1.2 Contents of the thesis

The contents of this thesis can be divided into two parts. Chapters 2, 3, and 4 discuss Algebraic nets from an algebraic perspective using the tools of category theory, while Chapter 5 tries to relate Algebraic nets to linear logic.

In chapter 2 we are mainly interested in giving the basic definitions of Algebraic nets. We take an axiomatic view of Algebraic nets; their semantics is defined through the unfolding to Petri nets. The basic aim of this chapter is to show that Algebraic nets are a good choice of high-level net formalism for our theoretical study. Algebraic nets are expressive enough in the sense that we can give semantics preserving translations into Algebraic nets for some more elaborate formalisms; but Algebraic nets are still simple enough, so that their definition is very short, and the syntax and semantics are very well separated.

Chapter 3 is concerned with giving the semantics of nets as universal constructions. We present two approaches. The first one is based on the structured

Figure 1.1: A Railway System as a CE-System (from (Genrich 1986)).

$$
\begin{array}{ll}
\texttt{sorts}: & \mathit{emtpytrack}, \mathit{occupiedtrack}, \mathit{train} \\
 & \\
\texttt{opns}: & 0,1,2,3,4,5,6 :\rightarrow \mathit{emptytrack} \\
 & \langle \_, \_ \rangle : \mathit{emptytrack} \times \mathit{train} \rightarrow \mathit{occupiedtrack} \\
 & a, b :\rightarrow \mathit{train} \\
 & f, g : \mathit{emptytrack} \rightarrow \mathit{emptytrack} \\
 & \\
\texttt{eqns}: & f(i) = i + 1 \text{ for } i = 0,1,2,3,4,5 \\
 & f(6) = 0 \\
 & g(i) = i - 1 \text{ for } i = 1,2,3,4,5,6 \\
 & g(0) = 6
\end{array}
$$

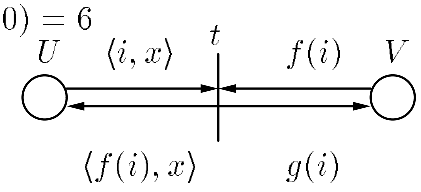


Figure 1.2: The Railway System as a high-level net.

transition system semantics of Corradini and Montanari (1992), while the second is based on the "Sheaves are objects" paradigm proposed by Goguen (1992). By defining the structured transition system semantics we are able to give a construction in which the free semantics of the low-level net together with the free semantics of the specification formalism define automatically the free semantics of the high-level net. Given this semantics, high-level nets are constructed by taking a tensor product of a low-level net class (in our case Petri nets) and a specification formalism (in our case many-sorted algebra). In the sheaf semantics a transition in a net is viewed as a sheaf and the net then gives a diagram of sheaves. The behavior of the net is given as the limit of this diagram. The interesting point about this semantics is that it is very much like a distributed implementation of Petri nets, where each transition is implemented on a single processor. The implementation of choice forces one to choose between an interleaving semantics, where only a single transition is allowed to fire, or a more concurrent semantics, where the resolution of conflict is done locally. The main contributions in this chapter are:

- The identification of high-level nets as graphs on monoids in a substitution system.
- The identification of interleaving and non-interleaving semantics as arising as the limit of a diagram of sheaves.

If in the previous chapter we have looked at high-level nets as formal systems of their own, in chapter 4 we return to the unfolding semantics. Since our original example motivated high-level nets as the folding of a low-level net, we want to look more thoroughly at this question. Is there some way in which this very intuitive notion of folding can be formalized? The main contributions of this chapter are:

- The formalization of the folding construction of Algebraic nets from Petri nets.
- A generalization of the folding construction to Algebraic nets.

In the final chapter we look at high-level nets from a logical point of view. It is well known, that the so called tensor-implication fragment of linear logic (Girard 1987a) can be seen as a logical axiomatization of Petri nets, and that the behavior of a Petri nets forms a quantale which is a model of linear logic (Engberg and Winskel 1994). In this chapter we extend this equivalence to high-level nets and a form of Predicate linear logic, by showing that there exists a functor from the category of Algebraic nets to the category of quantales.

We shall assume that the reader has a basic knowledge of algebraic specification, Net theory and category theory. The appendices contain a short introduction to category theory that fixes the notation, a review of the Petri nets are Monoids approach, and the relevant definitions from many-sorted algebra.

Chapter 2

Algebraic high-level nets

So called low-level net-classes are often criticized because of their unsuitability for modeling real applications. The main problem with these net-classes is that the resulting nets are huge. This problem has been aptly named the *football field* problem. One solution is to raise the level of abstraction and model only parts of the system. Another solution is to enrich the labeling of the nets to get a more concise and expressive formalism. Several such expressive formalisms have been proposed: Pr/T-nets (Genrich 1986), Coloured nets (Jensen 1986) and different flavors of Algebraic high-level nets (Reisig and Vautherin 1987, Dimitrovici, Hummert and Pétrucci 1990, Reisig 1991). Algebraic high-level nets are a combination of the algebraic specification method and net theory. The central idea is that the algebraic specification describes the data in the system, while net theory is used to specify the distribution and the movement of this data within the system.

In this work we will use the term Algebraic high-level net to mean the general class of high-level net formalisms based on algebraic specification. We will retain the term Algebraic net for the net-class defined here. We want to use the term Algebraic net, because the semantics in chapter 3 gives an algebraic structure on the nets.

As mentioned above, several different flavors of Algebraic high-level nets exist. The notion of Algebraic net presented by Dimitrovici and Hummert (1989) is slightly more complicated than ours. Our nets differ from the nets proposed by Reisig in (Reisig 1991) by allowing interpretations other than the initial. What this means is that we view multi-sets of terms essentially as an abbreviation for multiple arcs between transitions and places, while Reisig incorporates the notion of a multi-set into the specification of the abstract data type. The Algebraic nets we define are essentially the Algebraic nets of Reisig and Vautherin proposed in (Reisig and Vautherin 1987) in a categorical framework.

Although Dimitrovici et al. (1990) discuss Algebraic nets at length we feel that their treatment does not highlight the possible generalizations of Algebraic nets to other kinds of annotation formalism. Indeed In this chapter we will actually discuss two different Algebraic high-level formalisms, Algebraic nets and Order-sorted Algebraic nets. Their difference lies in the kind of algebraic specification formalism used. The theoretical strength of Algebraic nets comes from the fact that the two component formalisms are well separated, and that they are combined through a free functor that preserves most of the wanted properties. Indeed the proof of the main theorem about Order-sorted Algebraic nets can be borrowed directly from the proof for Algebraic nets. We also argue, that Algebraic nets form an expressive enough high-level formalism, by showing that Order-sorted Algebraic nets are equivalent to Algebraic nets.

The chapter is structured as follows. We start by defining Algebraic nets and their semantics. The main result of the first section is the fact that the semantics is a functor from the category of Algebraic nets to the category $\underline{\mathsf{PetriG}}$. In this section we also define some concepts that will be needed in the next chapters. Then we discuss the co-completeness of the category of Algebraic nets. Next we consider adding some features to the formalism. Finally we define Order-sorted Algebraic nets and prove that the category of Order-sorted Algebraic nets is equivalent to the category of Algebraic nets.

2.1 Algebraic nets

The aim of this section is to define a category of Algebraic nets and define their semantics in terms of Petri nets. The basic idea behind Algebraic nets is to label the arcs of the net with terms from an algebraic specification of a data type (Ehrig and Mahr 1985). Because algebraic specifications have a notion of implementation given in terms of so called Σ-algebras we are able to give a semantics for our high-level nets in terms of Petri nets. This semantics is the unfolding of the high-level net into a low-level net.

An Algebraic net (AN) consists of three parts, an abstract data-type specification Σ, a net that has arcs labeled with terms from this specification and a Σ-algebra that is the semantics of the specification. The basic definitions of universal algebra as it applies to the theory of abstract data-types is reviewed in appendix A.3.

Analogously to the definition of an ADT, we first define a "scheme" of nets. Then by fixing an interpretation we get an Algebraic net.

Definition 2.1.1
An *Algebraic net specification with equational signature* Σ (ANS) is a tuple

$$\langle S, \Sigma, EQ, X, T, P, \iota, o, sort \rangle$$

where:

- $\langle S, \Sigma, EQ \rangle$ is a Σ-presentation,
- X is a S-sorted set of variables,
- T is the set of transitions,
- P is the set of places,
- $\iota, o : T \to (P \times T_\Sigma(X))^\otimes$ are the input- and output-weight functions, and
- $sort : P \to S$ is the sort assignment.

$\square$

$(P \times T_\Sigma(X))^\otimes$ is the free monoid with the set $P \times T_\Sigma(X)$ as generators. In the sequel we will notate variables denoting transitions by $t, t_1, \ldots$ and variables denoting terms by $\hat{t}, \hat{t}_1, \ldots$. Let $Var(\hat{t})$ denote the set of variables of a term in $T_\Sigma(X)$. Then the set of variables of a term $m = \langle p, \hat{t} \rangle \otimes \langle p, \hat{t}_1 \rangle \otimes \ldots \otimes \langle p, \hat{t}_n \rangle \in (P \times T_\Sigma)^\otimes$ is defined by $Var(m) = \bigcup_{i=1}^n Var(\hat{t}_i)$. The set of variables of a transition is then given by $Var(t) = Var(\iota(t)) \bigcup Var(o(t))$.

In figure 2.1 we have a specification for the dining philosophers problem. The specification is divided into two parts, the first consisting of the algebraic specification of the net-inscriptions and the second consisting of the inscribed net. This particular specification contains 3 philosophers. The algebraic specification consists of three parts. The first part defines the sorts, or the types of individuals in the net, in this case philosophers and forks. The second part consists of the definition of the names and sorts of the operators in the specification. An operator with an empty argument list is called a constant and is used to represent individuals. In our case we have six individuals or constants, three philosophers and three forks. The specification also contains two functions l, and r, left and right respectively. They are used to map a philosopher to his left and right forks. This relationship is set up in the third section of the specification, through a set of equations. In this case the equations completely specify the behavior of the functions l and r. The arcs of the net are now annotated with terms formed from a set of variables and the operations declared in the `opns`-section of the specification[1]. The interpreta-

[1] To ease the pictorial presentation we have taken the liberty of using an isomorphism $\langle p, l(x) \rangle \otimes \langle p, r(x) \rangle \simeq \langle p, l(x) + r(x) \rangle$

`sorts`	:	$phil, fork$	
`opns`	:	ph_1, ph_2, ph_3	$:\to phil$
		f_1, f_2, f_3	$:\to fork$
		l	$: phil \to fork$
		r	$: phil \to fork$
`eqns`	:	$l(ph_i) = f_i$	
		$r(ph_i) = f_{i+1}$	for $i = 1, 2$
		$r(ph_3) = f_1$	

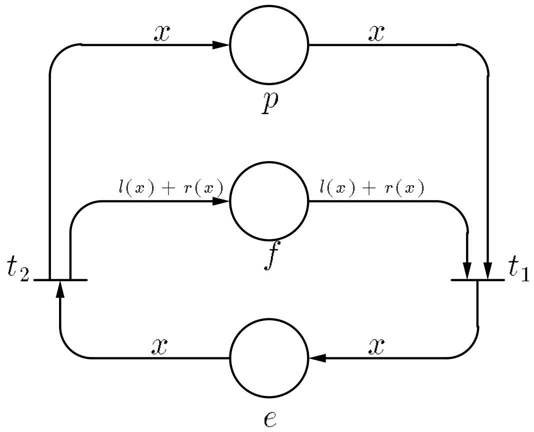


Figure 2.1: The specification NS of the dining philosophers problem.

tion of the input-arcs of transition t_1 read as follows. Given a philosopher x in place p, and a pair of forks $l(x)$ and $r(x)$ in place f, we can fire transition t_1 and move the philosopher to the place e. Suppose now that the place p is marked with philosopher ph_1. We then need forks $l(ph_1)$ and $r(ph_1)$ in place f for transition t_1 to be enabled. Using the set of equations `eqns`, the terms $l(ph_1)$ and $r(ph_1)$ can be evaluated and we see, that the forks we need are f_1, and f_2. If these tokens are available we say that transition t_1 *fires in mode* ph_1. This argument will be formalized below.

Since an ANS consists of two clearly separated parts, the net part and the algebraic specification part, it is advantageous to define a notion of morphism for ANS, by first defining a morphism that keeps the specification constant and only changes the net part.

Definition 2.1.2
A Σ-ANS-morphism $h : ANS \to ANS'$ is a pair $\langle h_T, h_P \rangle$ where $h_T : T \to T'$ and $h_P : P \to P'$ are functions s.t.:

$$\begin{aligned} h_P^{\S}(\iota(t)) &= \iota'(h_T(t)), \\ h_P^{\S}(o(t)) &= o'(h_T(t)), \quad \text{and} \\ sort'(h_P(p)) &= sort(p), \end{aligned}$$

where $h_P^{\S} : (P \times T_\Sigma(X))^{\otimes} \to (P' \times T_\Sigma(X))^{\otimes}$ is defined by $h_P^{\S}(\bigotimes_{i=1}^{n}\langle p, \hat{t}\rangle) = \bigotimes_{i=1}^{n}\langle h_P(p), \hat{t}\rangle$, with $p \in P$, $p' \in P'$. □

A Σ-ANS-morphism is a direct extension of a PetriG-morphism, that is the net morphisms are graph morphisms such that the place-morphisms respects the arc-weights (cf. figure 2.2). The reason for restricting ourselves to arc-weight preserving morphisms are two-fold: first we get a co-complete category of nets (cf. appendix A.2.2), and secondly we can define a notion of folding for P/T-nets (cf. chapter 4). It is easy to see the following result.

Proposition 2.1.3
Σ-ANS's and Σ-ANS-morphisms form a category Σ-ANS. □

A presentation morphism $h_\Sigma : \Sigma \to \Sigma'$ can be combined with a Σ-ANS-morphism to give an ANS-morphism. The signature morphism must preserve the arc-weights, and moreover it is required to preserve the sorts of the places. However we also need to specify how the set of variables in the specification are mapped. This amounts to the following definition.

Definition 2.1.4
An ANS-morphism $h : ANS \to ANS'$ is tuple $h = \langle h_\Sigma, h_X, h_T, h_P\rangle$ where $h_\Sigma : \Sigma \to \Sigma'$, $h_X : X \to X'$, $h_T : T \to T'$, and $h_P : P \to P'$ s.t.:

$$\begin{aligned} h_P^{\S}(\iota(t)) &= \iota'(h_T(t)), \\ h_P^{\S}(o(t)) &= o'(h_T(t)), \quad \text{and} \\ sort'(h_P(p)) &= h_S(sort(p)), \end{aligned}$$

where $h_P^{\S} : (P \times T_\Sigma(X))^{\otimes} \to (P' \times T_{\Sigma'}(X'))^{\otimes}$ is defined by

$$h^{\S}(\bigotimes_{i=1}^{n}\langle p, \hat{t}\rangle) = (\bigotimes_{i=1}^{n}\langle h_P(p), h_X^{\#}(\hat{t})\rangle),$$

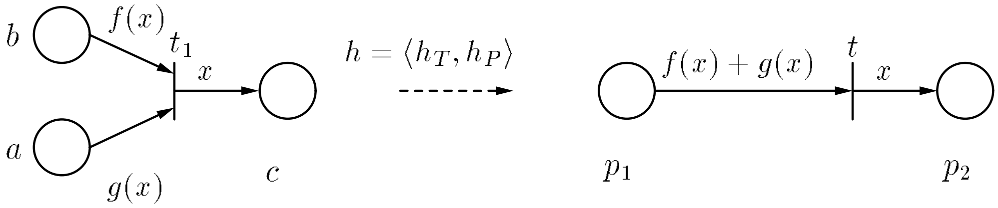
where $h_X^{\#} : T_\Sigma(X) \to T_{\Sigma'}(X')$ is the free extension of h_X along h_Σ. Moreover we require that h_X is an injective function. □

The condition that the morphism on variables is injective is required to get functoriality of the unfolding construction (cf. theorem 2.1.12).

Again we can form a category.

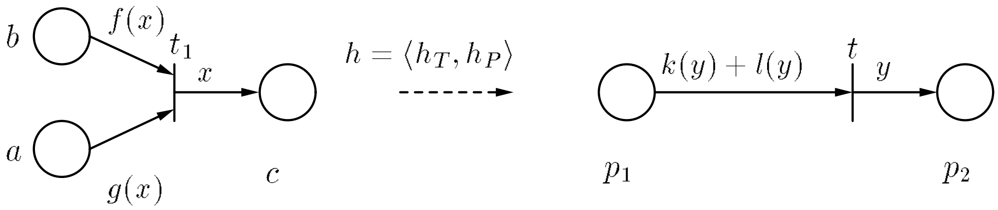
Proposition 2.1.5
ANS's and ANS-morphisms form the category ANS. □

Because there may be several interpretations (or implementations) of the abstract data-type specification, an ANS defines a *scheme* of nets. To give an



$$
\begin{array}{llcl}
h_T: & t_1 & \mapsto & t \\
h_P: & a & \mapsto & p_1 \\
 & b & \mapsto & p_1 \\
 & c & \mapsto & p_2
\end{array}
$$

Figure 2.2: A Σ-ANS-morphism.



$$
\begin{array}{llcl}
h_T: & t_1 & \mapsto & t \\
h_P: & a & \mapsto & p_1 \\
 & b & \mapsto & p_1 \\
 & c & \mapsto & p_2 \\
h_\Sigma: & f & \mapsto & k \\
 & g & \mapsto & l \\
h_X: & x & \mapsto & y
\end{array}
$$

Figure 2.3: A ANS-morphism.

interpretation of the specification we need to add a specific interpretation of the abstract data-type to our definition. This gives us the notion of an Algebraic net.

Definition 2.1.6
An *Algebraic net* AN is a pair $\langle ANS, A\rangle$ where ANS is an Algebraic net specification with signature Σ and A is a Σ-algebra. $\square$

If the algebra is not given, it is assumed to be $T_{\Sigma/\equiv}$. The default interpretation $T_{\Sigma/\equiv}$ is the quotient term algebra of the specification, which one usually has in mind when writing the specification. We can now define an AN-morphism as follows and then get a category of AN's.

Definition 2.1.7
Given Algebraic nets $AN = \langle ANS, A\rangle$ and $AN' = \langle ANS', A'\rangle$, an AN-morphism $h : AN \to AN'$ is a pair $\langle h_{ANS}, h_A\rangle$, where $h_{ANS} : ANS \to ANS'$ is an ANS-morphism, and $h_A : A \to A'|_{h_\Sigma}$ is a Σ-algebra homomorphism. $\square$

The map $h_A : A \to A'|_{h_\Sigma}$ is called a generalized morphism (cf. definition A.3.11).

Proposition 2.1.8
AN's and AN-morphisms form a category $\underline{\text{AN}}$. $\square$

The informal treatment of the dynamics of the AN given above can now be formalized by first defining a notion of marking.

Definition 2.1.9
A marking M in an Algebraic net is an element of $(P \times A)^{\otimes}$. If $A = T_\Sigma(X)$ and $Var(M) \neq \varnothing$ we say that the marking is *non-ground* or *abstract*. If there exists a substitution $\sigma : Var(t) \to A$, such that $(\iota(t); \sigma) \leq M$ with the pointwise ordering on monoids, we say that the transition t is enabled at M, and the new marking obtained by firing t is calculated by $M' = (M \ominus (\iota(t); \sigma)) \otimes (o(t); \sigma)$, where $a \ominus b$ is defined as the solution c to $b \otimes c = a$ if it exists, and the unit of the monoid otherwise. The traditional token game is obtained by requiring that the markings are ground. $\square$

In the philosophers example a reasonable initial marking would be $M_0 = \langle p, ph_1\rangle \otimes \langle p, ph_2\rangle \otimes \langle p, ph_3\rangle \otimes \langle f, f_1\rangle \otimes \langle f, f_2\rangle \otimes \langle f, f_3\rangle$. Now by substitution ph_1 for x in the input- and output-weights of t_1, we see that $(\iota(t_1); \{x \mapsto ph_1\}) = (\langle p, x\rangle \otimes \langle f, l(x)\rangle \otimes \langle f, r(x)\rangle); \{x \mapsto ph_1\} = \langle p, ph_1\rangle \otimes \langle f, f_1\rangle \otimes \langle f, f_2\rangle \leq M_0$, and thus we can apply the firing rule and obtain the new marking $M' = \langle p, ph_2\rangle \otimes \langle p, ph_3\rangle \otimes \langle f, f_3\rangle \otimes \langle e, ph_1\rangle$ representing the fact that philosopher ph_1 is now eating.

The token game above can also be interpreted as a token game on a low-level net, where expressions like $\langle p, ph_1\rangle$ are seen as names of places, and their occurrence in a tensor-expression as a transcription of the fact that the place is marked. Thus since the interpretation allows us to assign elements of the algebra A to the variables on the arcs, we can calculate the set of all possible assignments for the variables such that the transition would be enabled. An assignment $\mathrm{ass}_A^{\#} : Var(t) \rightarrow A$, on the free variables of the arcs connected to t, is called a *firing mode* of transition t. Each transition t in an AN is thought of as a shorthand for a set of transitions in a low-level net. This is the intuition that underlies the following *"unfolding"* construction.

To define the unfolding an Algebraic net we need the following auxiliary definition:

Definition 2.1.10
Let $\hat{t} = \hat{t}_1, \ldots, \hat{t}_n \in T_\Sigma(X)$.Given $\mathrm{ass}_A^{\#} : T_\Sigma(X) \rightarrow A$, define $\mathrm{ass}_A^* : (P \times T_\Sigma(X))^{\otimes} \rightarrow (P \times A)^{\otimes}$ by

$$\mathrm{ass}_A^*(\bigotimes_{i=1}^{n}\langle p, \hat{t}\rangle) = \bigotimes_{i=1}^{n}\langle p, \mathrm{ass}_A^{\#}(\hat{t})\rangle \ .$$

□

Definition 2.1.11
Given an Algebraic $AN = \langle ANS, A\rangle$ we can define a Petri net

$$\mathsf{Unf}(\langle ANS, A\rangle) = \langle T_{\mathsf{Unf}}, P_{\mathsf{Unf}}, \iota_{\mathsf{Unf}}, o_{\mathsf{Unf}}\rangle$$

as follows:

- $P_{\mathsf{Unf}} = \bigcup_{p\in P}\{\{p\} \times A_{sort(p)}\}$,
- $T_{\mathsf{Unf}} = \{\langle t, \mathrm{ass}_A^{\#}\rangle |\, \mathrm{ass}_A \in [Var(t) \rightarrow A], t \in T\}$,
- $\iota_{\mathsf{Unf}}(\langle t, \mathrm{ass}_A^{\#}\rangle) = \mathrm{ass}_A^*(\iota(t))$,
- $o_{\mathsf{Unf}}(\langle t, \mathrm{ass}_A^{\#}\rangle) = \mathrm{ass}_A^*(o(t))$.

□

For a place p in the high-level net we create a number of places $p', p'', p''', \ldots$ in the low-level net such that to every possible token x that may reside in place p there exists a place p' (ie. the pair $\langle p, x\rangle$) in the low-level net that represents the fact that x is in place p. Analogously for the transitions, for a transition t in the high-level net we have to create transitions in the low-level net that represent the fact that the transition fires with a specific assignment on the variables. It is an interesting question whether this unfolding has an inverse, and we shall return to this question in chapter 4.

An example will clarify the construction. Figure 2.4 gives an interpretation in the algebra:

$$A = (A, \alpha),$$

with

$$\begin{aligned} A &= \{\{f_1, f_2, f_3\}_{phil}, \{g_1, g_2, g_3\}_{fork}\} \text{ , and} \\ \alpha &= \{r, l, f_1, f_2, f_3, g_1, g_2, g_3\} \text{ .} \end{aligned}$$

We have omitted the obvious type declarations from the set of functions. The functions r, l have the following definition:

x	$r(x)$	$l(x)$
f_1	g_1	g_2
f_2	g_2	g_3
f_3	g_3	g_1

while the functions $f_1, f_2, f_3, g_1, g_2, g_3$ are the constant functions. The interpretation of NS in A is given in figure 2.4. The algebra A is the algebra $T_{\Sigma/\equiv}$. If one plays the standard token game for P/T-systems on this net it is easy to see, that it corresponds to the token game of the net in figure 2.1.

As mentioned previously, one usually thinks of the different interpretations of the abstract data-type specification as different implementations of the ADT. However there exist interpretations that cannot intuitively be seen as implementations of the ADT. One such interpretation is the terminal algebra $A^* = (\{phil, fork\}, \{\rightarrow fork, \rightarrow phil, fork \rightarrow phil\})$, where the sorts $fork$ and $phil$ have only one element, and the maps are the trivial maps. Interestingly it turns out, that this interpretation can be used to detect deadlocks efficiently in some cases. Figure 2.5 gives the unfolding of the philosophers net, with the interpretation A^*. The net shows just the amount of tokens consumed and produced by each transition, ie. transition t_1 takes one token from place p, 2 tokens from place f, and produces one token to place e. The resulting net is called the "skeleton". We shall discuss the skeleton of the net more thoroughly in section 4.3. An equivalent way of obtaining this interpretation is given by definition 2.1.15.

Since we are transforming an AN into a Petri net, the interesting question from a categorical point of view now is, whether this transformation is functorial. Indeed this is the case.

Theorem 2.1.12
Given an AN-morphism $h : AN \rightarrow AN'$ there exists a morphism $\mathsf{Unf}(f) : \mathsf{Unf}(AN) \rightarrow \mathsf{Unf}(AN')$, such that identities and composition are preserved, ie. $\mathsf{Unf} : \underline{\mathsf{AN}} \rightarrow \underline{\mathsf{PetriG}}$ is a functor.

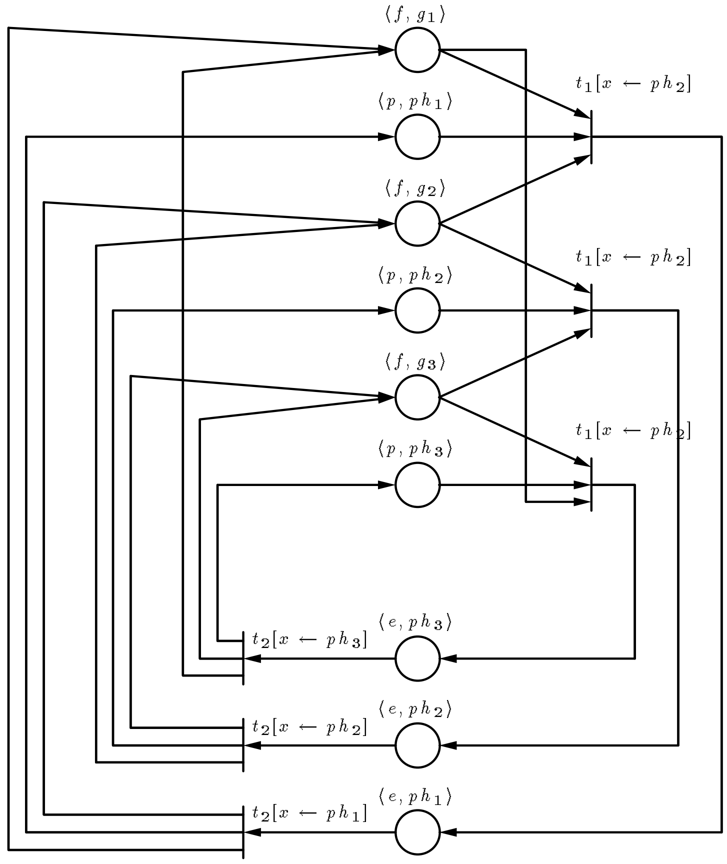


Figure 2.4: The interpretation of NS in A.

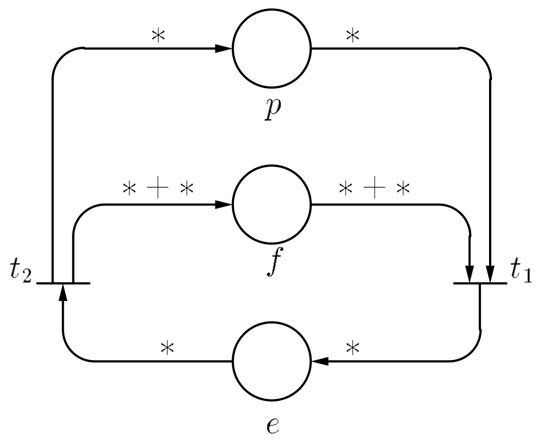


Figure 2.5: The interpretation of NS in A^*.

Proof:

1. $\mathsf{Unf}(f)$ is given by:

$$\begin{array}{llll} \mathsf{Unf}(f)_P: & P_{\mathsf{Unf}(AN)} \to P_{\mathsf{Unf}(AN')}: & \langle p, a\rangle \mapsto \langle h_P(p), h_A(a)\rangle \\ \mathsf{Unf}(f)_T: & T_{\mathsf{Unf}(AN)} \to T_{\mathsf{Unf}(AN')}: & \langle t, \mathrm{ass}_A^{\#}\rangle \mapsto \langle h_T(t), h_A(\mathrm{ass}_A^{\#})\rangle \,. \end{array}$$

2. That it is a PetriG-morphism is shown by the following tedious but straight-forward calculation:

$$\begin{array}{lll} \multicolumn{3}{l}{\mathsf{Unf}(f)_P^{\otimes}(\iota_{\mathsf{Unf}(AN)}(\langle t, \mathrm{ass}_A^{\#}\rangle))} \\ = & \mathsf{Unf}(f)_P^{\otimes}(\mathrm{ass}_A^{*}(\iota(t))) & (2.1.11) \\ = & \mathsf{Unf}(f)_P^{\otimes}(\bigotimes_{i=1}^{n}\langle \pi_1(\iota(t)_i), \mathrm{ass}_A^{\#}(\pi_2(\iota(t)_i))\rangle) & (2.1.10) \\ = & \bigotimes_{i=1}^{n}\langle h_P(\pi_1(\iota(t)_i)), h_A(\mathrm{ass}_A^{\#}(\pi_2(\iota(t)_i)))\rangle & (\text{def. of } \mathsf{Unf}(f)_P) \\ = & \mathrm{ass}_{A'}^{*}(\bigotimes_{i=1}^{n}\langle h_P(\pi_1(\iota(t)_i)), h_X^{\#}(\pi_2(\iota(t)_i))\rangle) & (2.1.10) \text{ and} \\ & & h_X^{\#}; \mathrm{ass}_{A'}^{\#} = \mathrm{ass}_A^{\#}; h_A \\ = & \mathrm{ass}_{A'}^{*}(h^{\S}(\bigotimes_{i=1}^{n}\langle \pi_1(\iota(t)_i), \pi_2(\iota(t)_i)\rangle)) & (\text{def. of } h^{\S}) \\ = & \mathrm{ass}_{A'}^{*}(h^{\S}(\iota(t))) & \\ = & \mathrm{ass}_{A'}^{*}(\iota'(h_T(t))) & (2.1.4) \\ = & \iota_{\mathsf{Unf}(AN')}(\langle h_T(t), h_A(\mathrm{ass}_A^{\#})\rangle) & (2.1.11) \\ = & \iota_{\mathsf{Unf}(AN')}(\mathsf{Unf}(f)_T(\langle t, \mathrm{ass}_A^{\#}\rangle)) & (\text{def. of } \mathsf{Unf}(f)_T) \end{array}$$

The above depends on the fact that $h_X^{\#}; \mathrm{ass}_{A'}^{\#} = \mathrm{ass}_A^{\#}; h_A$. However this is only true if the diagram

$$\begin{tikzcd} X \arrow[r, "ass_A"] \arrow[d, "h_X"'] & A \arrow[d, "h_A"] \\ X' \arrow[r, "ass_{A'}"] & A' \end{tikzcd}$$

commutes. A sufficient condition for this is that $h_X : X \to X'$ is injective (Padberg n.d.). To give a simple counterexample: let $X = \{x, y\}$, $X' = \{z\}$, and $A = A' = \{a, b\}$. If we now have $h_X : x \mapsto z, y \mapsto z$, $\mathrm{ass}_A : x \mapsto a, y \mapsto b$ and $\mathrm{ass}'_A : z \mapsto a$. Now $h_A(\mathrm{ass}_A(y)) = b$ while $\mathrm{ass}'_A(h_X(y)) = a$. The part for o is analogous.

3. The fact that the identities and composition are preserved is obvious.

□

It is worth commenting on the proof a bit. The interesting thing about the proof is that it never refers to the exact structure of the assignment notion in use. Actually, if we look at the definition of the unfolding, we see that the only reference to the structure of the assignments is given in the definition of T_{Unf}

as the set $\{\langle t, \mathrm{ass}_A^{\#}\rangle | \, \mathrm{ass}_A \in [Var(t) \to A], t \in T\}$, where actually we see that the assignment is an element of a set of functions. So it would seem plausible to suggest that any notion of specification formalism that defines a notion of assignment might be usable as a basis for defining a high-level net class. We shall indeed investigate this question more thoroughly in this chapter, and we shall see, that we can really "reuse" the proof of theorem 2.1.12.

The idea of "taking the skeleton" of an Algebraic net was originally conceived by Vautherin (1987). Intuitively this amounts to forgetting the specific annotations on the arcs and just keeping the weights. The Algebraic net is thus transformed into a Petri net with the same structure as the Algebraic net. Formally this amounts to taking the terminal algebra $\perp$ as the interpretation of the specification Σ.

Definition 2.1.13
The *underlying net* functor $\mathsf{U_N} : \underline{\mathsf{AN}} \to \underline{\mathsf{PetriG}}$ maps and Algebraic net to is underlying Petri net. It is given by the assignment $\mathsf{U_N}(AN) = \langle T_U, P_U, \iota_U, o_U\rangle$ with

- $T_U = T$,
- $P_U = P$,
- $\iota_U(t) = \iota(t); \mathrm{ass}_\perp^*$
- $o_U(t) = o(t); \mathrm{ass}_\perp^*$

The morphisms part of the definition is given through theorem 2.1.12. □

Proposition 2.1.14
The morphism $f : \mathsf{Unf}(AN) \to \mathsf{U_N}(AN)$ is unique.

Proof:
Follows from the fact that $\mathsf{U_N}(\langle ANS, A\rangle) \simeq \mathsf{Unf}(\langle ANS, \perp\rangle)$ and that $\perp$ is the terminal algebra. □

Each Petri net can be viewed as an Algebraic net if we think of each arc in the Petri net as being annotated with a constant c. This can be formalized in the following construction that defines a functor $\mathsf{G} : \underline{\mathsf{PetriG}} \to \underline{\mathsf{AN}}$, that maps each $\underline{\mathsf{PetriG}}$-net N to the corresponding $\underline{\mathsf{AN}}$-net $G(N)$.

Definition 2.1.15
The functor $\mathsf{G} : \underline{\mathsf{PetriG}} \to \underline{\mathsf{AN}}$, that transforms each $\underline{\mathsf{PetriG}}$-net N into an $\underline{\mathsf{AN}}$-net $G(N)$ is given by

$$G(N) = \langle\langle \{*\}, \{c :\to *\}, \phi, \phi, T_G, P_G, \iota_G, o_G, sort_G\rangle, \{*\}, \{c :\to *\}\rangle$$

where $T_G = T_N$, $P_G = P_N$, and the weighting functions $\iota_G, o_G : T_G \to (P_G \times \{c\})$ are given by the equations

$$\begin{array}{rcl} \iota_G(t) & = & i(\iota_N(t)) \\ o_G(t) & = & i(o_N(t)), \end{array}$$

where i is the injection $i : P^{\otimes} \to (P \times \{c\})^{\otimes}$. □

The functor is characterized by the following lemma.

Lemma 2.1.16
$\mathsf{Unf}(\mathsf{G}(N)) \simeq N$.

Proof:
We first show that the sets of places and transitions are isomorphic.

$$\begin{array}{rcl} \mathsf{Unf}(\mathsf{G}(N))_P & = & \bigcup_{p \in P} \{\{p\} \times A_{sort_G}\} \\ & = & \bigcup_{p \in P} \{\{p\} \times \{*\}\} \\ & = & P \times \{*\} \\ & \simeq & P \end{array}$$

$$\begin{array}{rcl} \mathsf{Unf}(\mathsf{G}(N))_T & = & \{\langle t, \mathrm{ass}_A^{\#}\rangle | t \in T, \mathrm{ass}_A^{\#} \in [Var(t) \to A]\} \\ & = & \{\langle t, \phi\rangle | t \in T\} \\ & \simeq & T \end{array}$$

The input and output weight functions are then shown to be isomorphic through the following equations:

$$\begin{array}{rcl} \iota_{\mathsf{Unf}}(\langle t, \phi\rangle) & = & \iota_N(t) \\ o_{\mathsf{Unf}}(\langle t, \phi\rangle) & = & o_N(t) \end{array}$$

□

2.2 Co-completeness of AN

The aim of this section is to prove that AN is co-complete. The notion of co-limit is a categorical notion used to describe the "pasting together" of mathematical structures. The use of co-limits to combine smaller specification to larger ones is a technique first applied in the specification language CLEAR (Burstall and Goguen 1977, Burstall and Goguen 1990). It is now a standard technique in the theory of algebraic specifications. On the other hand, in categorical approaches to net-theory co-limits have played little role until recently. This is maybe in part due to the negative result of Winskel in (Winskel 1985) on the non-existence of coproducts in the category of Place/Transition-nets with initial marking. In the paper by Meseguer and Montanari (Meseguer and Montanari 1990) limits and co-limits are mentioned, but no interesting examples except the standard product and coproduct constructs are discussed. The recent paper by Dimitrovici et al. (1990) contains a thorough discussion of co-limits in their category of Algebraic high-level nets. Also the dissertation of Hummert (Hummert 1989) contains interesting applications of co-limits to nets. He defines a notion of net module and studies the compatibility of the notion with invariants and parameterized data structures. In this chapter when we give a proof of co-completeness of the category $\underline{\mathrm{AN}}$ we want to see, what are the features that are actually needed for the proof.

Theorem 2.2.1
The category $\underline{\mathrm{AN}}$ is co-complete.

Proof:
Essentially the proof depends on the fact that the categories $\underline{\mathsf{Set}}$ and $\underline{\mathsf{Alg}}$ are co-complete.

- $\underline{\mathrm{AN}}$ has as initial object the AN $\langle\varnothing,\varnothing,\varnothing,\varnothing,\varnothing,\varnothing,\varnothing\rangle$.
- *co-products*: Co-products are just disjoint union. Let

$$AN_i = \langle S_i, \Sigma_i, EQ_i, X_i, T_i, P_i, \iota_i, o_i, sort_i, A_i\rangle$$

for $i = 1, 2$. Define

$$\begin{aligned} AN_1 + AN_2 \quad = \quad & \langle S_1 \uplus S_2, \Sigma_1 \uplus \Sigma_2, EQ_1 \uplus EQ_2, X_1 \uplus X_2, \\ & T_1 \uplus T_2, P_1 \uplus P_2, \iota_1 + \iota_2, o_1 + o_2, sort_1 + sort_2, \\ & A_1 + A_2\rangle \end{aligned}$$

The morphisms $\iota_1+\iota_2 : T_1 \uplus T_2 \to (P_1 \times T_{\Sigma_1}(X_1))^{\otimes} \uplus (P_2 \times T_{\Sigma_2}(X_2))^{\otimes}$ and $o_1 + o_2 : T_1 \uplus T_2 \to (P_1 \times T_{\Sigma_1}(X_1))^{\otimes} \uplus (P_2 \times T_{\Sigma_2}(X_2))^{\otimes}$ are calculated componentwise. Clearly the injections $in_i : AN_i \to AN_1 + AN_2$ are $\underline{\mathrm{AN}}$-morphisms.

$$\begin{tikzcd} AN_1 \arrow[r, "in_1"] \arrow[dr, "f_1"'] & AN_1 + AN_2 \arrow[r, "in_2"] \arrow[d, "\varphi"] & AN_2 \arrow[dl, "f_2"] \\ & AN & \end{tikzcd}$$

The universal arrow $\varphi = \langle \varphi_\Sigma, \varphi_X, \varphi_T, \varphi_P, \varphi_A \rangle$ is defined by $\varphi_\Sigma = f_{\Sigma_1} + f_{\Sigma_2}, \varphi_X = f_{X_1} + f_{X_2}, \varphi_T = f_{T_1} + f_{T_2}, \varphi_P = f_{P_1} + f_{P_2}, \varphi_A = f_{A_1} + f_{A_2}$. The fact that it is universal follows from the fact that its components are universal since both $\underline{\mathsf{Alg}}$ and $\underline{\mathsf{Set}}$ are co-complete.

- *co-equalizers*: Co-equalizers are also calculated component-wise. Let $f : AN_1 \to AN_2$ and $g : AN_1 \to AN_2$ be $\underline{\mathsf{AN}}$-morphisms. Their co-equalizer $q : AN_2 \to AN$ in the diagram

$$AN_1 \underset{g}{\overset{f}{\rightrightarrows}} AN_2 \xrightarrow{q} AN$$

  is given by $q_\Sigma = coeq(f_\Sigma, g_\Sigma)$ in $\underline{\mathsf{Sig}}$, $q_X = coeq(f_X, g_X)$, $q_P = coeq(f_P, g_P)$ and $q_T = coeq(f_T, g_T)$ in $\underline{\mathsf{Set}}$, and $q_A = coeq(f_A, g_A)$ in $\underline{\mathsf{Alg}}$. Again, because the components are universal arrows it follows that $\underline{\mathsf{AN}}$ has co-equalizers.

- Because $\underline{\mathsf{AN}}$ has initial object, co-products and co-equalizers it is co-complete.

□

So the co-completeness of the category $\underline{\mathsf{AN}}$ depends on two factors:

1. The category $\underline{\mathsf{Set}}$ is co-complete, and
2. the categories of algebras and signatures are co-complete.

Thus we can claim that replacing many-sorted algebra with any specification formalism that is co-complete, we get a co-complete category of high-level nets.

2.3 Conditions on transitions

Usually it is useful to have more features in the formalism, where these features are meant to increase the expressive power the formalism. One such feature is the addition of conditions on transitions (cf. (Reisig 1991) sec. 10.3). The idea is to constrain the set of possible assignments by setting up a predicate on the variables of the transition. Here a predicate is seen as an operation with sort *bool*. However as we shall see, adding conditions on transitions mainly consists of adding "syntactic sugar".

Definition 2.3.1
An *Algebraic net with conditions* (ANSC) is a tuple

$$\langle S, \Sigma, EQ, X, T, P, \iota, o, sc, sort, A \rangle$$

where:

- $\langle S, \Sigma, EQ \rangle$ is a Σ-presentation, with a sort *bool* and a constant $\mathbf{true} :\to bool$
- X is a S-sorted set of variables,
- T is the set of transitions,
- P is the set of places,
- $\iota, o : T \to (P \times T_\Sigma(X))^{\otimes}$ are the input- and output-weight functions,
- $c : T \to T_{\Sigma, bool}(X)$,
- $sort : P \to S$ is the sort assignment, and
- A is a Σ-algebra.

□

Clearly every AN is and ANSC with no conditions, and the notion of marking is the same. The firing rule is modified, so that the transition is enabled iff

$$(i(t); \mathrm{ass}_A^{\#}) \leq M \wedge A \models \mathrm{ass}_A^{\#}(c(t)) = \mathbf{true},$$

ie. if the assignment satisfies the predicate. This gives rise to the notion of a consistent transition assignment (Ehrig, Padberg and Ribero 1992).

Definition 2.3.2
The set of *consistent transition assignments* is given by:

$$CT = \{\langle t, \mathrm{ass}_A^{\#} \rangle \parallel t \in T : A \models \mathrm{ass}_A^{\#}(c(t)) = \mathbf{true}\}$$

□

A morphism of Algebraic nets with conditions is essentially an AN-morphism that is required to respect the set of conditions.

Definition 2.3.3
Given Algebraic nets with conditions $ANSC = \langle ANS, A \rangle$ and $ANSC' = \langle ANS', A' \rangle$, an ANSC-morphism $h : ANSC \to ANSC'$ is a pair $\langle h_{ANS}, h_A \rangle$, where $h_{ANS} : ANS \to ANS'$ is an ANS-morphism, and $h_A : A \to A'|_{h_\Sigma}$ is a Σ-algebra morphism, such that $f_T; c' = c; f_\Sigma$. The presentation morphism h_Σ is also required to preserve the boolean part of the presentation. □

Proposition 2.3.4
Algebraic nets with conditions and ANSC-morphism for a category <u>ANSC</u>.

□

The unfolding of the Algebraic nets with conditions is now defined analogously to definition 2.1.11 using the set CT.

Definition 2.3.5
For each Algebraic net with side-conditions $ANSC$ we have a P/T-net $\mathsf{Unf}(ANSC) = \langle P_F, T_F, \iota_F, o_F \rangle$ given by

- $P_F = \bigcup_{p \in P} \{\{p\} \times A_{sort(p)}\}$,
- $T_F = CT$,
- $\iota_F(\langle t, \mathrm{ass}_A^{\#} \rangle) = \mathrm{ass}_A^{*}(\iota(t)$,
- $o_F(\langle t, \mathrm{ass}_A^{\#} \rangle) = \mathrm{ass}_A^{*}(o(t))$.

□

Proposition 2.3.6
The construction defines a functor $\mathsf{Unf} : \underline{\mathsf{ANSC}} \rightarrow \underline{\mathsf{PetriG}}$.

Proof:
The proof is exactly the same as the proof of theorem 2.1.12. □

Adding conditions does not add to the expressive power of the Algebraic net formalism. As we shall see in chapter 4, each Petri net can be represented by a unique Algebraic net. However we can also give a syntactic transformation that transforms each Algebraic net with conditions to an Algebraic net without conditions, albeit we now must restrict the possible initial markings. The intuition of the transformation is presented in figure 2.6.

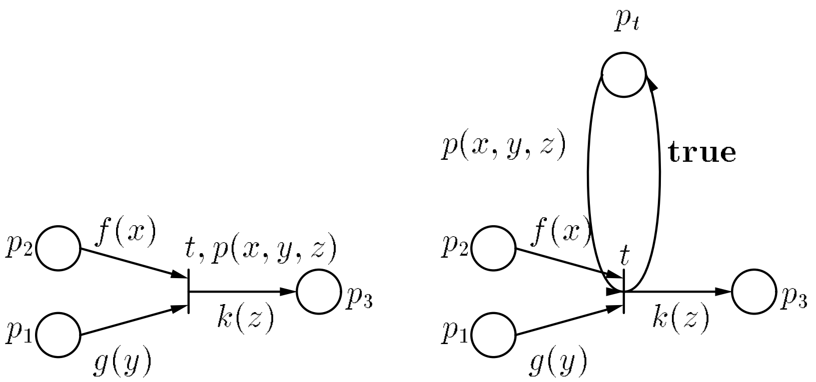


Figure 2.6: Transforming conditions on transitions.

The idea is simply to add a place p_t with sort *bool* to each transition t in the original net. The arc from p_t to t is labeled with the predicate and the arc from t to p_t with **true**. If p_t is now initially labeled with **true**, the transition t is enabled in the transformed net iff it is in the original net.

The construction can be formalized as follows:

Proposition 2.3.7
Given an Algebraic net with conditions $AN_c = \langle S, \Sigma, EQ, X, T, P, \iota, o, sc, sort\rangle$ we can construct an algebraic net $AN = \langle S, \Sigma, EQ, X, T, P', \iota', o', sort'\rangle$ such that

$$M \xrightarrow{t} M' \quad \text{in } AN \text{ iff } \quad M_c \xrightarrow{t} M_c' \quad \text{in } AN_c,$$

where $M = M_c \bigotimes_{t\in T} \langle p_t, \mathbf{true}\rangle$.

Proof:
The net is given by $P' = P \bigcup \{p_t\}_{t\in T}$, $\iota'(t) = \iota(t) \otimes \langle p_t \times c(t)\rangle$, $o'(t) = o(t) \otimes \langle p_t \times \mathbf{true}\rangle$. The fact that the set of reachable states are isomorphic is easily seen through the observation that both AN_c and AN have the same sets of consistent transition assignments. □

2.4 Order-sorted Algebraic nets

Order-sorted algebra has been proposed by Goguen and Meseguer (1992) as a generalization of many-sorted algebra to support abstract data-types with multiple inheritance, polymorphism and overloading, exception handling, and partial operations.

One of the motivations of Order-sorted algebra is that it allows for the elegant treatment of error conditions in the theory of abstract datatypes. The standard example concerns the treatment of preventing the taking of $top(emptystack)$ in the stack. Several solutions to prevent this from happening have been presented. Using order-sorted algebra the problem is solved by defining a type `NeStack` of non-empty stacks that is a sub-sort of the sort of stacks and then defining the operation top to be of sort `NeStack` $\rightarrow$ `Elt`. Thus the term $top(emptystack)$ will never be an element of $T_{\Sigma/\equiv}$. An analogous situation may arise in an Algebraic net. Consider the net in figure 2.7. The place p_1 contains a queue. The queue is read through transition t_2 and written through transition t_1. However we want to prevent t_2 from firing when the queue is empty. Using Algebraic nets with conditions we can add the condition $nonempty(x)$ to the transition t_2 to prevent this. Using Order-sorted algebra we require the sort of the variable x to be `NeQueue`. Thus we see, that in the case of Algebraic nets the use of Order-sorted algebra and conditions on transitions are in some sense equivalent. The claim we would like to make is, that the specification with conditions on transitions is clearer. It is easier to see directly from the net that there is the possibility of an empty queue in p_1. As we shall see below, we are also able to show that any Order-sorted net can be transformed into an equivalent Algebraic net. There is however a theoretical interest in going through the definitions because it again shows clearly what the structure of a high-level net is.

sorts :	$Elt, Queue$
opns :	$enqueue : Queue \times Elt \rightarrow Queue$
	$head : Queue \rightarrow Elt$
	$tail : Queue \rightarrow Queue$
	$emptyqueue :\rightarrow Queue$
	$non - empty : Queue \rightarrow bool$
eqns :	
	$nonempty(emptyqueue) = false$
	$nonempty(enqueue(Q, E)) = true$
	$\vdots$

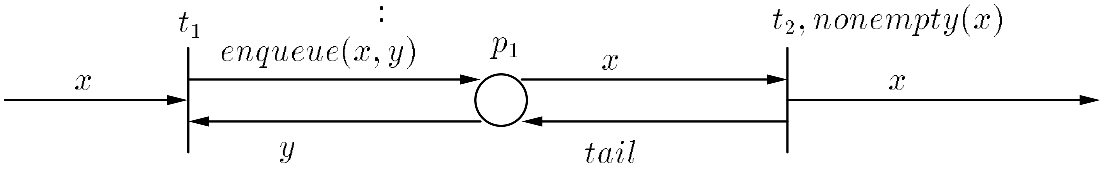


Figure 2.7: An Algebraic net modeling a queue.

Definition 2.4.1
An *Order-sorted Algebraic net specification with equational signature* Σ (OS-ANS) is a tuple

$$\langle S, \leq, \Sigma, EQ, X, T, P, \iota, o, sort \rangle$$

where:

- $\langle S, \leq, \Sigma, EQ \rangle$ is an order-sorted Σ-presentation,
- X is a S-sorted set of variables,
- T is the set of transitions,
- P is the set of places,
- $\iota, o : T \rightarrow (P \times T_\Sigma(X))^{\otimes}$ are the input- and output-weight functions, and
- $sort : P \rightarrow S$ is the sort assignment.

$\square$

Definition 2.4.2
An OSANS-morphism $h : OSANS \rightarrow OSANS'$ is tuple $h = \langle h_\Sigma, h_T, h_P \rangle$ where $h_\Sigma : \Sigma \rightarrow \Sigma'$ is an order-sorted signature morphism, $h_X : X \rightarrow X'$, $h_T : T \rightarrow T'$, and $h_P : P \rightarrow P'$ s.t.:

$$h_P^{\#}(\iota(t)) \quad = \quad \iota'(h_T(t)),$$

$$\begin{array}{rcl} h_P^{\#}(o(t)) & = & o'(h_T(t)), \text{ and} \\ sort'(h_P(p)) & = & h_S(sort(p)) \end{array}$$

where $h_P^{\#} : (P \times T_\Sigma(X))^{\otimes} \to (P' \times T_\Sigma(X)')^{\otimes}$ is defined by

$$h^{\#}(\bigotimes_{i=1}^{n}\langle p, \hat{t}\rangle) = (\bigotimes_{i=1}^{n}\langle h_P(p), h_\Sigma^{\#}(\hat{t})\rangle),$$

and h_X is an injective function. □

Proposition 2.4.3
OSANS's and OSANS-morphisms form the category <u>OSANS</u>. □

Definition 2.4.4
An *Order-sorted Algebraic net* OSAN is a pair $\langle OSAN, A\rangle$ where OSAN is an order-sorted Algebraic net specification with signature Σ and A is a Σ-algebra. □

If the algebra is not given, it is assumed to be $T_{\Sigma/\equiv}$.

Definition 2.4.5
Given order-sorted Algebraic nets $OSAN = \langle OSAN, A\rangle$ and $OSAN' = \langle OSAN', A'\rangle$, an OSAN-morphism $h : OSAN \to OSAN'$ is a pair $\langle h_{OSAN}, h_A\rangle$, where $h_{OSAN} : OSAN \to OSAN'$ is an OSAN-morphism, and $h_A : A \to A'|_{h_\Sigma}$ is a Σ-algebra morphism. □

Proposition 2.4.6
OSAN's and OSAN-morphisms form a category <u>OSAN</u>. □

Like for Algebraic nets with and without side-conditions, we can define an unfolding of an Order-sorted Algebraic net $OSAN$.

Definition 2.4.7
Given an Order-sorted Algebraic net $OSAN = \langle OSANS, A\rangle$ we can define a Petri net

$$\mathsf{Unf}(\langle OSANS, A\rangle) = \langle T_{\mathsf{Unf}}, P_{\mathsf{Unf}}, \iota_{\mathsf{Unf}}, o_{\mathsf{Unf}}\rangle$$

as follows:

- $P_{\mathsf{Unf}} = \bigcup_{p \in P} p \times A_{sort(p)}$,
- $T_{\mathsf{Unf}} = \{\langle t, \mathrm{ass}_A^{\#}\rangle \mid \mathrm{ass}_A \in [Var(t) \to A], t \in T\}$,
- $\iota_{\mathsf{Unf}}(\langle t, \mathrm{ass}_A^{\#}\rangle) = \mathrm{ass}_A^{*}(\iota(t))$,
- $o_{\mathsf{Unf}}(\langle t, \mathrm{ass}_A^{\#}\rangle) = \mathrm{ass}_A^{*}(o(t))$.

□

Again this defines a functor.

Proposition 2.4.8
The construction defines a functor $\mathsf{Unf} : \underline{\mathsf{OSAN}} \to \underline{\mathsf{PetriG}}$.

Proof:
The proof is exactly the same as the proof of theorem 2.1.12. □

Because there exists an equivalence of categories between the category of order-sorted algebras with signature Σ $\underline{\mathsf{OSAlg}_\Sigma}$ and the category of algebras $\underline{\mathsf{Alg}_{\Sigma^\#,J}}$ of a specification $\Sigma^\#$ together with a set of conditional equations J (Goguen and Meseguer 1992, Theorem 4.2), it is interesting to see whether this equivalence would lift to the level of Algebraic nets and Order-sorted Algebraic nets.

Theorem 2.4.9
Given an Order-sorted Algebraic net $OSAN$ with coherent signature Σ, there exists an Algebraic net AN with signature $\Sigma^\#$ that satisfies the conditional equations J, with the same set of places and transitions, and such that $\mathsf{Unf}(OSAN) \simeq \mathsf{Unf}(AN)$, and conversely.

Proof:
The latter direction is trivial, since every many-sorted algebra is an order-sorted algebra with an empty ordering. For the former direction the idea is to transform each order-sorted signature Σ into a many-sorted signature $\Sigma^\#$ as shown in theorem A.3.16. Since an order-sorted Σ-algebra is up to isomorphism a $\Sigma^\#$-algebra that satisfies the conditional equations J, it is easy to see, that the corresponding Petri nets will be isomorphic. □

What the construction does is, it transforms an order-sorted Algebraic net with signature Σ, and interpretation A, into an Algebraic net with signature $\Sigma^\#$ and interpretation $A^\#$, where $A^\#$ is the order-sorted algebra A seen as a many-sorted algebra. So the Algebraic net obtained through the transformation does not have $T_{\Sigma^\#/\equiv}$ as its interpretation. The interesting thing about this result is the way in which a result on the annotation formalisms lifts to a result on the corresponding high-level net formalisms.

2.5 Final remarks

The main points in this chapter could be summed up as follows:

Algebraic nets are powerful enough: This was shown by two examples. In the first example we added a new feature to the net-part of the specification formalism (conditions on transitions), while in the second example we added an extension to the algebraic specification formalism (sub-sorts). In both cases we were able to give a behaviorally equivalent algebraic net without these extensions.

Algebraic nets are a clean formalism: By this we mean that the different conceptual components of the formalism, net-theory, multisets and algebraic specification, are connected through very clean interfaces, basically just a free functor. This leads to a very simple co-completeness proof. Also the fact that the semantics is based on the idea of substitution allows us to transfer the proof of the functoriality of the unfolding semantics to the two extended net formalisms presented in this chapter.

As far as answering the question of what a high-level net is, we have not yet come very far. However, in this chapter we have identified several components of a high-level net formalism, and we have identified the notion of substitution as the main component in the definition of the semantics. In the next chapter we will look at the notion of substitution more thoroughly and show how to base the definition of a high-level net formalism on the notion of substitution.

Chapter 3

Semantics

In this chapter we shall look at two very different semantics for nets. The first one, structured transition system semantics, is a highly algebraic semantics, that relies heavily on the notion of *free* construction. The idea here is that in many cases there is an a algebraic structure on the states of the transition system that can be lifted to the level of the transitions through a free construction. The computations can then be obtained by freely completing the given structured transition system to a category. The second semantics, sheaf semantics, is a more concrete semantics. Instead of relying on a free construction, where the semantics is determined by the free functor, sheaf theory allows us to implement the semantics that we want. Thus we are able to give both an interleaving semantics and a non-interleaving semantics. The advantage of the structured transitions system semantics is that since it relies on the free construction, once we have set up some basic requirements, we get the semantics "for free": it is generated by a set of rules. On the other hand sheaf semantics gives us an alternative view of a transition. It is seen as an object that establishes a relational constraint on the behavior of the tokens flowing through it. A net is a network of such objects and the behavior of the net is again an object that satisfies all the constraints of the component objects at the same time. Common to both semantics is that the behavior is obtained through a universal construction, the free construction in the case of structured transition system semantics and the limit in the case of sheaf semantics.

The main contributions in this chapter are:

- The identification of high-level nets as graphs on monoids in a substitution system.
- The identification of the tensor product of sketches as the fundamental building block of high-level nets.

- The identification of interleaving and non-interleaving semantics as arising as the limit of a diagram of sheaves.

3.1 Structured transition system semantics

The aim of this section is to give a general description of high-level nets as structured transition systems (STS). Structured transition systems together with a method for obtaining algebraic semantics for formalisms were proposed by Corradini and Montanari (1992) in the context of semantics of logic programs. The method is a generalization of the algebraic treatment of Petri nets proposed by Meseguer and Montanari (1990). In the method one distinguished between three description levels in the formalism:

1. the level of heterogeneous graphs, whose nodes have more structure than their arcs,
2. the level of structure transition systems, where the structure on the nodes has been lifted to the arcs, and
3. the level of models of structured transition systems, where the structure is extended to the computations of the transition system.

These levels are connected by free functors. The power of the method lies in the fact that these free functors are essentially extensions of the left adjoint of the forgetful functor that relates the structure of the nodes to the structure on the arcs, so that once these structures have been identified the rest is obtained "for free".

In this section we will first review the basic steps for obtaining a structured transition system from a Petri net. This construction is then used to obtain a structured transition system of the unfolded Algebraic net. Subsequently we discuss the general method and finally apply it to Algebraic nets. We have omitted those proofs that can be found in (Corradini and Montanari 1992).

3.1.1 Structured transition systems through unfolding

In this subsection we show, by reviewing the approach of Meseguer and Montanari (1990) how a structured transition system can be obtained through unfolding.

A transition system is often defined as a relation $R \subseteq S \times S$ on a set of states S. The transition system contains only the information that a state s' is reachable from state s. However, quite often it is useful to attach some more information to the state transitions, like what action was executed in the state transition. This leads to the following definition of a transition system.

Definition 3.1.1
A *transition system* is a graph $TS = \langle T, S, \delta_0, \delta_1 \rangle$, where T is the set of transitions and S is the set of states, and $\delta_0, \delta_1 : T \to S$ are defined by $\delta_0(t) = s$, $\delta_1(t) = s'$ iff transition t leads from state s to state s'. □

Computations in a transition system TS are sequences of transition steps. This corresponds to composition of arrows in the graph. Since there exists a standard free functor $\mathsf{F} : \underline{\mathsf{Graph}} \to \underline{\mathsf{Cat}}$, that takes each graph to the free category generated by it, it would suggest, that the natural structure on the computations in a TS is the free category generated by its graph.

However, as observed by Meseguer and Montanari (1990) and reviewed in appendix A.2, in the case of Petri nets, the states do not form a set. Instead they are elements of a free commutative monoid, making our transition system a heterogeneous graph, with transitions and states living in different categories. In this case it is not directly obvious what the composition of a transition should be. For example if we have transitions $t_1 : a \otimes b \to c, t_2 : d \to e, t_3 : c \otimes e \to f$ neither t_1 nor t_2 compose directly with t_3. The monoidal structure on the states can however be lifted to the transitions, so that we can form a transition $t_1 \otimes t_2 : a \otimes b \otimes d \to c \otimes e$ (t_1 and t_2 in parallel) and then compose this transition with t_3 to obtain $(t_1 \otimes t_2); t_3 : a \otimes b \otimes d \to e$. It turns out that we also need to add "idle" transitions for each place $t_p : p \to p$ to represent the fact that "nothing happens". Then we are able to compose $t_4 : a \to b$ with $t_5 : a \otimes b \to c$ as $(t_a \otimes t_4); t_5 : a \otimes a \xrightarrow{t_a \otimes t_4} a \otimes b \xrightarrow{t_5} c$. Consequently we actually lift the monoidal structure to a reflexive monoidal structure.

In categorical terms, this amounts to the existence of free functors F_{PGCM} : $\underline{\mathsf{Petri}} \to \underline{\mathsf{CMonRPetri}}$ and $\mathsf{F}_{CMCP} : \underline{\mathsf{CMonRPetri}} \to \underline{\mathsf{CatPetri}}$. The first functor lifts the monoidal structure on the places to the transitions, and adds reflexivity (the idle transitions), while the second functor completes the reflexive structured graph to the category of computations of the Petri net. The category $\underline{\mathsf{CMonRPetri}}$ is the structured transition system induced by a Petri net, while $\underline{\mathsf{CatPetri}}$ is the category of models.

It is now possible to obtain a structured transition system of an Algebraic net by first unfolding it, and then by applying the free functor $\mathsf{F} : \underline{\mathsf{Petri}} \to \underline{\mathsf{CatPetri}}$. The construction of the transition system is done using the rules in figure 3.1. The rules describe how a Petri net is completed to a free category. Rule 3.4 lifts the monoidal structure on the states to the transitions, while the rest of the rules complete the Petri net into a category. Rule 3.1 adds identities, rule 3.2 adds the transitions as arrows, and rule 3.3 defines the

$$\frac{u \text{ in } P^{\otimes}}{t_u : u \to u \text{ in } \mathsf{T}[N]} \tag{3.1}$$

$$\frac{\iota(t) = u, o(t) = v \text{ and } t \text{in } N}{t : u \to v \text{ in } \mathsf{T}[N]} \tag{3.2}$$

$$\frac{t_1 : u \to v, t_2 : v \to w \text{ in } \mathsf{T}[N]}{t_1; t_2 : u \to w \text{ in } \mathsf{T}[N]} \tag{3.3}$$

$$\frac{t : u \to v, t' : u' \to v' \text{ in } \mathsf{T}[N]}{t \otimes t' : u \otimes u' \to v \otimes v' \text{ in } \mathsf{T}[N]} \tag{3.4}$$

Figure 3.1: The rules that define a functor $\mathcal{T}[\_\_] : \underline{\mathsf{Petri}} \to \underline{\mathsf{CatPetri}}$.

composition of two arrows. Using these rules we essentially describe the computations of an Algebraic net through the computations of its unfolded net. However it is possible to generalize the construction to an arbitrary category of places and transitions by using the techniques of internal category theory. This allows for a direct description of the computations of the Algebraic net.

3.1.2 Structured transition systems directly

To generalize the Petri nets are monoids approach of Meseguer and Montanari (1990) we first need to introduce some tools. These tools are internal category theory and the theory of sketches. The use of internal category theory is motivated by its suggestive nature, while the theory of sketches provides some powerful tools for proving the existence of left adjoints. Once these tools have been introduced we will present the 4 steps that make up the methodology of Corradini and Montanari (1992), and apply these steps to the definition of Algebraic nets.

Internal categories and sketches

The basic idea in the method proposed by Corradini and Montanari (1992) is the notion of internalization. Internalization means that for example instead of looking at the category of graphs, we look at the objects in $\underline{\mathsf{Set}}$ that are graphs. Once this abstraction has been done we can transfer the concept of graph to other categories.

Consider the definition of a graph $G = \langle A, N, \delta_0, \delta_1 \rangle$. A graph consists of two sets, the arcs and nodes, and two functions $\delta_0, \delta_1 : A \to N$, that live in

Set, so that a graph can be considered as a diagram $A \overset{\delta_0}{\underset{\delta_1}{\rightrightarrows}} N$ in Set, also known as the "graph of graphs". Similarly a graph morphism correspond to the commutativity of the diagram:

$$\begin{tikzcd} A \arrow[r, shift left, "\delta_0"] \arrow[r, shift right, "\delta_1"'] \arrow[d, "f_A"'] & N \arrow[d, "f_N"] \\ A' \arrow[r, shift left, "\delta_0"] \arrow[r, shift right, "\delta_1"'] & N' \end{tikzcd}$$

in Set.

By abstracting away from Set we can define graph and graph morphisms in any category.

Definition 3.1.2
Given a category $\underline{C}$, a tuple $g = \langle c_0, c_1, \delta_0, \delta_1 \rangle$ is an *internal graph* of $\underline{C}$, iff:

- $c_0, c_1 \in |\underline{C}|$, and
- $\delta_0, \delta_1 : c_0 \to c_1 \in \mathbf{Mor}_{\underline{C}}$.

Let $g = \langle c_0, c_1, \delta_0, \delta_1 \rangle$ and $g' = \langle c_0', c_1', \delta_0', \delta_1' \rangle$ be internal graphs. An *internal graph morphism* $f : g \to g'$ is a pair of morphisms $f_0 : c_o \to c_0'$ and $f_1 : c_1 \to c_1'$ in $\underline{C}$ such that the diagram

$$\begin{tikzcd} c_0 \arrow[r, shift left, "\delta_0"] \arrow[r, shift right, "\delta_1"'] \arrow[d, "f_0"'] & c_1 \arrow[d, "f_1"] \\ c_0' \arrow[r, shift left, "\delta_0"] \arrow[r, shift right, "\delta_1"'] & c_1' \end{tikzcd}$$

commutes. Graph($\underline{C}$) is the category of internal graphs of $\underline{C}$.

A reflexive graph is a graph where for every node n there exists an arc $id_n : n \to n$. This can be internalized as follows. An *internal reflexive graph* in $\underline{C}$ is a tuple $r = \langle c_0, c_1, \delta_0, \delta_1, id \rangle$, where $r = \langle c_0, c_1, \delta_0, \delta_1 \rangle$ is an internal graph, and $id : c_1 \to c_0$ is a morphism of $\underline{\mathsf{C}}$, such that the diagram

$$\begin{tikzcd}
& c_1 \arrow[dl, "id_{c_1}"'] \arrow[d, "id"'] \arrow[dr, "id_{c_1}"] & \\
c_1 & c_0 \arrow[l, "\delta_0"'] \arrow[r, "\delta_1"] & c_1
\end{tikzcd}$$

commutes. An *internal reflexive graph morphism* is an internal graph morphisms such that the diagram

$$\begin{tikzcd}
c_0 \arrow[r, "f_0"] \arrow[d, "id"'] & c_0' \arrow[d, "id'"'] \\
c_1 \arrow[r, "f_1"] & c_1'
\end{tikzcd}$$

commutes. $\underline{\mathsf{RGraph}}(\underline{\mathsf{C}})$ is the category of internal reflexive graphs of $\underline{\mathsf{C}}$.

$\square$

Proposition 3.1.3
If $\underline{\mathsf{C}}$ is a category with binary co-products, then the forgetful functor $\mathsf{U} : \underline{\mathsf{RGraph}}(\underline{\mathsf{C}}) \to \underline{\mathsf{Graph}}(\underline{\mathsf{C}})$ has a free adjoint $\mathsf{F} : \underline{\mathsf{Graph}}(\underline{\mathsf{C}}) \to \underline{\mathsf{RGraph}}(\underline{\mathsf{C}})$ defined as $\mathsf{F}(\langle c_0, c_1, \delta_0, \delta_1 \rangle) = \langle c_0 + c_1, c_1, \delta_0 + id_{c_1}, \delta_1 + id_{c_1}, in_2 \rangle$. $\square$

Let us try to make this same kind of analysis for the definition of a category. If a category $\underline{\mathsf{C}}$ is small, then the collection of morphisms $\mathbf{Mor}_{\underline{\mathsf{C}}}$ is a set, and the collection of objects form a set $|\underline{\mathsf{C}}|$. This means that we have a pair of functions $\delta_0, \delta_1 : \mathbf{Mor}_{\underline{\mathsf{C}}} \to |\underline{\mathsf{C}}|$ that specify the domain and codomain of each morphism. Two morphisms can be composed iff $\delta_1(f) = \delta_0(g)$. We can then define a function *comp* that has as domain the set $\{(f, g) \mid \delta_1(f) = \delta_0(g)\}$, and as codomain $\mathbf{Mor}_{\underline{\mathsf{C}}}$. This is exactly the pullback of δ_0 and δ_1, which exists in $\underline{\mathsf{Set}}$ since it is complete. An object c together with the above functions is called an *internal category*. In the category of small categories, the category $\underline{\mathsf{Set}}$ can be though of as the universe of discourse. So the general notion is given by the following definition.

Definition 3.1.4
Given a category $\underline{\mathsf{C}}$ a tuple $c = \langle c_0, c_1, \delta_0, \delta_1, comp, id \rangle$ is an internal category object of the category $\underline{\mathsf{C}}$ iff $\langle c_0, c_1, \delta_0, \delta_1, id \rangle$ is an internal reflexive graph in $\underline{\mathsf{C}}$, and $comp : c_0 \times_0 c_0 \to c_0$, where $c_0 \times_0 c_0$ is the pullback of δ_0, δ_1, such that the following diagrams commute.

$$\begin{tikzcd} c_0 \times_0 c_0 \arrow[r, "comp"] \arrow[d, "\pi_1"'] & c_0 \arrow[d, "\delta_0"] \\ c_0 \arrow[r, "\delta_0"] & c_1 \end{tikzcd} \qquad \begin{tikzcd} c_0 \times_0 c_0 \arrow[r, "comp"] \arrow[d, "\pi_2"'] & c_0 \arrow[d, "\delta_1"] \\ c_0 \arrow[r, "\delta_1"] & c_1 \end{tikzcd}$$

comp1 comp2

$$\begin{tikzcd} c_0 \times_0 c_0 \times_0 c_0 \arrow[r, "comp \times_0 id"] \arrow[d, "id \times_0 comp"'] & c_0 \times_0 c_0 \arrow[d, "comp"] \\ c_0 \times_0 c_0 \arrow[r, "comp"] & c_0 \end{tikzcd}$$

assoc

$$\begin{tikzcd} c_0 \times_0 c_1 \arrow[r, "id_{c_0} \times id"] \arrow[dr, "\pi_1"'] & c_0 \times_0 c_0 \arrow[d, "comp"] \\ & c_0 \end{tikzcd} \qquad \begin{tikzcd} c_1 \times_0 c_0 \arrow[r, "id \times id_{c_0}"] \arrow[dr, "\pi_2"'] & c_0 \times_0 c_0 \arrow[d, "comp"] \\ & c_0 \end{tikzcd}$$

neutr1 neutr2

The definition of an *internal functor* is given as an internal reflexive graph morphism that also preserves composition. $\underline{\mathfrak{Cat}}(\underline{\mathrm{C}})$ is the category of internal categories of $\underline{\mathrm{C}}$. □

As can be seen from the definition, a prerequisite for the existence of internal category object is the existence of suitable pullbacks. In this subsection all our categories will be equipped with pullbacks. The concept of internalization can be extended to other categorical notions like natural transformations and adjunctions. These notions have recently become an important tool in the study of models of the second order lambda-calculus (Asperti and Longo 1991).

If we examine the treatment of Petri nets in the previous subsection, we notice that Petri nets are not internal graphs, since the nodes and transitions live in different categories, but that their completion to objects $\underline{\mathsf{CMonPetri}}$ are graphs in $\underline{\mathsf{CMon}}$, the category of commutative monoids. Now since there exists a forgetful functor $\mathsf{U}_{CMSe} : \underline{\mathsf{CMon}} \to \underline{\mathsf{Set}}$, the notion of internal graph can be generalized as follows.

Definition 3.1.5
Let $\underline{\mathsf{C}}$ and $\underline{\mathsf{B}}$ be two categories such that the forgetful functor $\mathsf{U} : \underline{\mathsf{C}} \to \underline{\mathsf{B}}$ exists. Then $g = \langle b, c, \delta_0, \delta_1 \rangle$ is a *heterogeneous graph with arcs in* $\underline{\mathsf{B}}$ *and nodes in* $\underline{\mathsf{C}}$ iff $b \in |\underline{\mathsf{B}}|$ and $c \in |\underline{\mathsf{C}}|$ and $\delta_0, \delta_1 : b \to \mathsf{U}(c)$, are morphisms in $\underline{\mathsf{B}}$.

A *heterogeneous graph morphism* $f : \langle b, c, \delta_0, \delta_1 \rangle \to \langle b', c', \delta_0', \delta_1' \rangle$ is a pair f_0, f_1, where $f_0 : b \to b'$ is an arrow in $\underline{\mathsf{B}}$ and $f_1 : c \to c'$ is an arrow in $\underline{\mathsf{C}}$, such that the following diagrams commute:

$$\begin{tikzcd}
b \arrow[r, "f_0"] \arrow[d, "\delta_0"'] & b' \arrow[d, "\delta_0'"] \\
\mathsf{U}(c) \arrow[r, "\mathsf{U}(f_1)"] & \mathsf{U}(c')
\end{tikzcd}
\qquad
\begin{tikzcd}
b \arrow[r, "f_0"] \arrow[d, "\delta_1"'] & b' \arrow[d, "\delta_1'"] \\
\mathsf{U}(c) \arrow[r, "\mathsf{U}(f_1)"] & \mathsf{U}(c')
\end{tikzcd}$$

The category of heterogeneous graphs with arcs in $\underline{\mathsf{B}}$ and nodes in $\underline{\mathsf{C}}$ is denoted by $\underline{\mathsf{Graph}}(\underline{\mathsf{B}}, \underline{\mathsf{C}})$. □

Essentially a heterogeneous graph $\langle b, c, \delta_0, \delta_1 \rangle$ is an internal graph $\langle b, \mathsf{U}(c), \delta_0, \delta_1 \rangle$ in $\underline{\mathsf{B}}$, but the morphisms must preserve the richer structure on nodes. The category of Petri nets $\underline{\mathsf{Petri}}$ is the category $\underline{\mathsf{Graph}}(\underline{\mathsf{Set}}, \underline{\mathsf{FCMon}})$

Recall that the commutative monoidal structure on the places could be lifted to the transitions along the free functor $\mathsf{F}_{SeCM} : \underline{\mathsf{Set}} \to \underline{\mathsf{CMon}}$. The general structure of this process is described in the following proposition.

Proposition 3.1.6
Let $\underline{\mathsf{Graph}}(\underline{\mathsf{B}}, \underline{\mathsf{C}})$ be the category of heterogeneous graphs, such that the forgetful functor $\mathsf{U} : \underline{\mathsf{C}} \to \underline{\mathsf{B}}$ has a free adjoint $\mathsf{F} : \underline{\mathsf{B}} \to \underline{\mathsf{C}}$, and let $\mathsf{U}' : \underline{\mathsf{Graph}}(\underline{\mathsf{C}}) \to \underline{\mathsf{Graph}}(\underline{\mathsf{B}}, \underline{\mathsf{C}})$ be the forgetful functor induced by U, which forgets about the structure of arcs. Then U' has a free adjoint $\mathsf{F}' : \underline{\mathsf{Graph}}(\underline{\mathsf{B}}, \underline{\mathsf{C}}) \to \underline{\mathsf{Graph}}(\underline{\mathsf{C}})$ given by

$$\mathsf{F}'(\langle b, c, \delta_0, \delta_1 \rangle) = \langle \mathsf{F}(b), c, \mathsf{F}(\delta_0); \varepsilon_c, \mathsf{F}(\delta_1); \varepsilon_c \rangle,$$

on objects, and by

$$\mathsf{F}'(\langle f_0, f_1 \rangle) = \langle \mathsf{F}(f_0), f_1 \rangle$$

on morphisms, where $\varepsilon : \mathsf{F}; \mathsf{U} \to 1_{\underline{\mathsf{C}}}$ is the co-unit of the adjunction $\langle \mathsf{F}, \mathsf{U}, \varphi \rangle$.

□

We will also need the notion of internal commutative monoid.

Definition 3.1.7
Given a category $\underline{\mathsf{C}}$ with product $\times$ and terminal object $\perp$, a tuple $c = \langle c, \mu, \eta, \gamma \rangle$ is called a commutative monoid in $\underline{\mathsf{C}}$ iff $c \in |\underline{\mathsf{C}}|$ and $\mu : c \times c \to c$, $\eta : \perp \to c$, and $\gamma : c \times c \to c \times c$ are morphisms in $\underline{\mathsf{C}}$ such that the following diagrams commute:

$$\begin{tikzcd}
c \times c \times c \arrow[r, "\mu \times id"] \arrow[d, "id \times \mu"'] & c \times c \arrow[d, "\mu"] \\
c \times c \arrow[r, "\mu"] & c
\end{tikzcd}
\qquad
\begin{tikzcd}
\perp \times c \arrow[dr, "snd"'] & c \times c \arrow[l, "\eta \times id"'] \arrow[r, "id \times \eta"] & c \times \perp \arrow[dl, "fst"] \\
& c &
\end{tikzcd}$$

$$\begin{tikzcd}
\perp \times c \arrow[rr, "\gamma"] \arrow[dr, "snd"'] & & c \times \perp \arrow[dl, "fst"] \\
& c &
\end{tikzcd}
\qquad
\begin{tikzcd}
c \times c \arrow[r, "\gamma"] \arrow[dr, "id"'] & c \times c \arrow[d, "\gamma"] \\
& c \times c
\end{tikzcd}$$

The map μ is called the operation of the monoid, the map η the unit, and the map γ is the symmetry. A morphism $f : \langle c, \mu, \eta \rangle \to \langle c', \mu', \eta' \rangle$ of monoids is an arrow $f : c \to c'$ such that

$$\mu; f = (f \times f); \mu' : c \times c \to c',$$

$$\eta; f = \eta' : e \to c',$$

and

$$\gamma; (f \times f) = (f \times f); \gamma' .$$

The commutative monoids of a category $\underline{\mathsf{C}}$ form a category $\underline{\mathsf{CMon}}(\underline{\mathsf{C}})$, with a forgetful functor $\mathsf{U} : \underline{\mathsf{Mon}}(\underline{\mathsf{C}}) \to \underline{\mathsf{C}}$ defined by the assignment $\langle c, \mu, \eta \rangle \mapsto c$. □

For example in $\underline{\mathsf{Set}}$ the monoid objects are the ordinary monoids.

The above definitions all rely on the existence of left ("free") adjoints. Also below we shall need the existence of a left adjoint $\mathsf{C}_{\underline{\mathsf{C}}} : \underline{\mathsf{Rgraph}}(\underline{\mathsf{C}}) \to \underline{\mathsf{Cat}}(\underline{\mathsf{C}})$. Usually it is quite cumbersome to prove the existence of adjoints. However, fortunately all the internal categories presented above can be seen as models of so called LE-sketches, and the theory of sketches contains the necessary tools to prove the existence of the left adjoints.

From a Computer Science point of view, sketches can be considered as an alternative approach to the specification of abstract datatypes. The reason for the development of the theory of sketches is that there are many important mathematical entities that are not equationally specifiable. The prime example is the theory of fields. It is not possible to specify equationally that division is defined only on a subset of the domain. In this case it turns out however that one can express this very elegantly with a co-limit. Thus allowing limits and co-limits in our specification we are able to expand the expressive power of our specification formalism. As an other example there exists a sketch of Horn theories, which are known not to be equational in the sense of Lawvere (1963). The classical algebraic approach, as initiated by Lawvere (1963), is however a special case of the sketch approach. The standard reference on the basic theory of sketches is chapter 4 of (Barr and Wells 1985).

Since we can view the operators in a signature as arrows in a graph, and limits are also effectively expressed as graphs the definition of a sketch is the following.

Definition 3.1.8
A LE-sketch S is a 4-tuple $S = \langle G, U, D, C \rangle$, where G is a graph, $U : N \rightarrow A$ is a function which takes each node $n \in N$ to an arrow $n \rightarrow n$ of A, D is a class of diagrams in G, and C is a class of cones in G. A sketch morphism $S \rightarrow S'$ is a graph morphism $h : G \rightarrow G'$ such that $U; h = h; U'$, and every diagram in D is taken to a diagram in D', and every cone in C is taken to a cone in C'. □

Because in our case we do not need to specify structures with co-limits, we work with so called LE-sketches which is short for Left Exact which means that it has all finite limits. In the definition the graph is the set of all arrows of the specification together with the arrows from all the diagrams and cones. The diagrams correspond to the equations, while the limits are used to describe subsets.

Translating the definition of an internal category (definition 3.1.4) into a sketch gives us as the graph the "graph of graphs" together with all the arrows in the diagrams. The class of diagrams is the set of 5 diagrams, and the class of cones contains only the pullback $c_0 \times_0 c_0$, describing the set of composable arrows, a subset of the set of arrows.

Conversely we can given a category $\underline{C}$ construct a sketch $\mathcal{SK}_{\underline{C}}$, that we can use to define the model of a sketch.

Definition 3.1.9
Given a category $\underline{C}$, the *underlying sketch* $\mathcal{SK}_{\underline{C}} = \langle G, U, D, C \rangle$ of $\underline{C}$ has as graph the underlying graph of $\underline{C}$, as U the map that picks out the identity

arrows of $\underline{C}$, as D the class of commutative diagrams of $\underline{C}$, and as C the class of limit cones of $\underline{C}$.

A *model* for a sketch S in a category $\underline{C}$ is a sketch morphism from S into $\mathcal{SK}_{\underline{C}}$. A model forces all the diagrams of the sketch to commute and all the cones of the sketch to be limit cones.

The models in $\underline{C}$ form a category $\mathsf{Mod}_{\underline{C}}(S)$ with natural transformations as morphisms. $\mathsf{Mod}(S)$ is the category of models in $\underline{\mathrm{Set}}$. □

The fact that allows us to switch back and forth between the view as internal concepts and the view as models of sketches is that for example in the case of graphs $\underline{\mathsf{Graph}}(\underline{\mathrm{Set}}) = \mathsf{Mod}(\mathcal{SK}_{\underline{\mathsf{Graph}}}) = \underline{\mathsf{Graph}}$. That is the category of graphs in $\underline{\mathrm{Set}}$ is the same as the category of models of the sketch of graphs, which is the same as the category of graphs. Indeed we shall often write $\underline{C}(\underline{D})$ for $\mathsf{Mod}_{\underline{D}}(\mathcal{SK}_{\underline{C}})$.

The existence of left adjoints is ensured by the following theorem.

Theorem 3.1.10
Let A and B be LE-sketches. If $h : A \to B$ is a sketch morphism, then the induced morphism $h^* : \mathsf{Mod}(B) \to \mathsf{Mod}(A)$ has a left adjoint $h_\# : \mathsf{Mod}(A) \to \mathsf{Mod}(B)$. □

However this theorem is not enough to prove the existence of the left adjoint $\mathsf{C}_{\underline{C}} : \underline{\mathsf{Rgraph}}(\underline{C}) \to \underline{\mathsf{Cat}}(\underline{C})$. The reason is that it is not clear whether $\underline{\mathsf{Rgraph}}(\underline{C})$ and $\underline{\mathsf{Cat}}(\underline{C})$ are models of LE-sketches. However, it turns out that both $\underline{\mathsf{Rgraph}}(\underline{C})$ and $\underline{\mathsf{Cat}}(\underline{C})$ are models of $\mathsf{Mod}(\mathcal{SK}_{\underline{\mathsf{RGraph}}} \otimes \mathcal{SK}_{\underline{C}})$ and $\mathsf{Mod}(\mathcal{SK}_{\underline{\mathsf{Cat}}} \otimes \mathcal{SK}_{\underline{C}})$ respectively, where $\otimes$ is a tensor product on sketches defined by Gray (1987).

Definition 3.1.11
Let $S = \langle G, U, D, C\rangle$ and $S' = \langle G', U', D', C'\rangle$ be sketches, with $G = \langle G_0, G_1, \delta_0, \delta_1\rangle$ and $G' = \langle G'_0, G'_1, \delta'_0, \delta'_1\rangle$. Then the *tensor product* $S \otimes S' = \langle G'', U'', D'', C''\rangle$ of the sketches is defined by:

1. $G''_0 = G_0 \times G'_0$,
2. $G''_1 = G_1 \times \{U'(s') | s' \in G'_0\} \bigcup G'_1 \times \{U(s) | s \in G_0\}$,
3. $U''(\langle s, s'\rangle) = \langle U(s), U'(s')\rangle$,
4. The set of diagrams D'' consists of, for every node s in G_0 a copy of all the diagrams in D', indexed by s, and for every node s' in G'_0 a copy of all the diagrams in D, indexed by s', together with for each arrow

$h : s_1 \to s_2$ in G_1 and $h' : s'_1 \to s'_2$ in G'_1 a diagram:

$$\begin{tikzcd}
\langle s_1, s'_1 \rangle \arrow[r, "h_{s'_1}"] \arrow[d, "h'_{s_1}"] & \langle s_2, s'_1 \rangle \arrow[d, "h'_{s_2}"] \\
\langle s_1, s'_2 \rangle \arrow[r, "h_{s'_2}"] & \langle s_2, s'_1 \rangle
\end{tikzcd}$$

5. The set of cones C'' consists of, for every node s in G_0 a copy of all the cones in C', indexed by s, and for every node s' in G'_0 a copy of all the cones in C, indexed by s'

□

The underlying idea of the tensor product is very simple. One just takes the product of the graphs of the sketches and the union of the diagrams and cones and adjoins a set of diagrams that mainly state, that the operations from G commute with those from G'.

The importance of this construction is shown by the following theorem

Theorem 3.1.12
$\mathsf{Mod}(A \otimes B) \simeq \mathsf{Mod}_{\mathsf{Mod}(B)}(A)$.

Proof:
See (Gray 1987, Proposition 3.6(ii)). □

The theorem states that the category of models of a tensor product of sketches is equivalent to the category of models of A in the category of models of B, ie. the structure of A is added "on top" of the structure of B.

As can also be seen from the definition the tensor product is commutative which implies the following theorem.

Theorem 3.1.13
$\underline{\mathsf{C}}(\underline{\mathsf{D}}) \simeq \underline{\mathsf{D}}(\underline{\mathsf{C}})$. □

What this means is that it does not matter in which order the structure is added "on the top".

We can now finally state the main theorem.

Theorem 3.1.14 (Theorem I.2.1 (Corradini and Montanari 1992))
1. If $\underline{\mathsf{C}}$ is the category of models in $\underline{\mathsf{Set}}$ of a left-exact sketch, then the forgetful functor $\underline{\mathsf{Cat}}(\underline{\mathsf{C}}) \to \underline{\mathsf{Rgraph}}(\underline{\mathsf{C}})$ has a left adjoint $\mathsf{C}_{\underline{\mathsf{C}}} : \underline{\mathsf{Rgraph}}(\underline{\mathsf{C}}) \to \underline{\mathsf{Cat}}(\underline{\mathsf{C}})$.

2. Let $\underline{C}$ be a category with all finite limits and co-limits, with all ω-co-limits, such that finite limits commute with ω-co-limits, then the forgetful functor $\underline{\mathsf{Cat}}(\underline{C}) \to \underline{\mathsf{Rgraph}}(\underline{C})$ has a left adjoint $\mathsf{C}_{\underline{C}} : \underline{\mathsf{Rgraph}}(\underline{C}) \to \underline{\mathsf{Cat}}(\underline{C})$.

Proof:
We shall only prove the first part of the theorem, since this is the part that we will use. We have an obvious sketch morphism $h : \mathcal{SK}_{\underline{\mathsf{RGraph}}} \to \mathcal{SK}_{\underline{\mathsf{Cat}}}$, which induces a morphism $\mathcal{SK}_{\underline{\mathsf{RGraph}}} \otimes \mathcal{SK}_{\underline{C}} \to \mathcal{SK}_{\underline{\mathsf{Cat}}} \otimes \mathcal{SK}_{\underline{C}}$. By theorem 3.1.10 this again induces a functor $\mathsf{Mod}(\mathcal{SK}_{\underline{\mathsf{Cat}}} \otimes \mathcal{SK}_{\underline{C}}) \to \mathsf{Mod}(\mathcal{SK}_{\underline{\mathsf{RGraph}}} \otimes \mathcal{SK}_{\underline{C}})$, that has a left adjoint $\mathsf{Mod}(\mathcal{SK}_{\underline{\mathsf{RGraph}}} \otimes \mathcal{SK}_{\underline{C}}) \to \mathsf{Mod}(\mathcal{SK}_{\underline{\mathsf{Cat}}} \otimes \mathcal{SK}_{\underline{C}})$. The statement follows because $\mathsf{Mod}(\mathcal{SK}_{\underline{\mathsf{RGraph}}} \otimes \mathcal{SK}_{\underline{C}}) \simeq \mathsf{Mod}_{\mathsf{Mod}(\mathcal{SK}_{\underline{\mathsf{RGraph}}})}(\mathcal{SK}_{\underline{C}}) \simeq \underline{\mathsf{Rgraph}}(\underline{C})$ and $\mathsf{Mod}(\mathcal{SK}_{\underline{\mathsf{Cat}}} \otimes \mathcal{SK}_{\underline{C}}) \simeq \mathsf{Mod}_{\mathsf{Mod}(\mathcal{SK}_{\underline{\mathsf{Cat}}})}(\mathcal{SK}_{\underline{C}}) \simeq \underline{\mathsf{Cat}}(\underline{C})$. $\square$

We now have all the relevant tools to apply the methodology of Corradini and Montanari (1992). Let us first briefly recall the main ideas of the methodology. It consists of the following steps (Corradini and Montanari 1992, (p.70)):

1. First determine the "natural" structure of the states and the transitions in the high-level net and the corresponding morphisms. The category is of the form $\underline{\mathsf{Graph}}(\underline{B}, \underline{C})$, ie. a heterogeneous graph with states in $\underline{C}$ and transitions in $\underline{B}$. In general there is a forgetful functor $\mathsf{U} : \underline{C} \to \underline{B}$, together with a free functor $\mathsf{F} : \underline{B} \to \underline{C}$.

2. Next define the category of transition systems that models the dynamics of the high-level net. This category is the category $\underline{\mathsf{Rgraph}}(\underline{C})$ of reflexive internal graphs in $\underline{C}$. Under suitable conditions we have a forgetful functor $\mathsf{U} : \underline{\mathsf{Rgraph}}(\underline{C}) \to \underline{\mathsf{Graph}}(\underline{B}, \underline{C})$ together with a free functor $\mathsf{F} : \underline{\mathsf{Graph}}(\underline{B}, \underline{C}) \to \underline{\mathsf{Rgraph}}(\underline{C})$.

3. Then define the category of models for transitions systems. This is done using the free functor $\mathsf{C}_{\underline{C}} : \underline{\mathsf{Rgraph}}(\underline{C}) \to \underline{\mathsf{Cat}}(\underline{C})$ from the reflexive internal graphs to the internal categories of $\underline{C}$.

4. Finally we need to distinguish the intended models of the high-level net N as a subcategory $\underline{\mathsf{Mod}}(N)$ of $\underline{\mathsf{Cat}}(\underline{C})$.

The advantage of this method is that once one has determined the structure of the category $\underline{C}$ then one obtains "for free" the other structures by going along the free functors. Thus in our case it is sufficient to determine the heterogeneous graph of high-level nets. However for step 3 to succeed, we must ensure the existence of the free functor $\mathsf{C}_{\underline{C}} : \underline{\mathsf{Rgraph}}(\underline{C}) \to \underline{\mathsf{Cat}}(\underline{C})$, for which we need theorem 3.1.14.

Algebraic nets as graphs on monoidal substitution systems

To apply the method we must first discover the structure on the places and transitions. Corradini and Montanari (1992) develop as their application a structured transition system semantics of logic programs. The idea is that predicates correspond to states, and clauses to transitions. Interestingly this same idea has been used by Murata and Zhang (1988) to give a high-level net description of logic programs. Thus it would seem plausible to take these ideas as a starting point. Indeed we shall do so, and show that high-level nets arise as graphs in the category of monoids on a substitution system.

The first step is to represent an Algebraic net as a heterogeneous graph. Let us start by trying to identify the structure on the states. Recall from definition 2.1.9 that a marking is a subset of $(P \times A)^{\otimes}$. Let us for the moment take $A = T_{\Sigma}(X)$. The first transformation that we make is that we add the set of places P to the signature Σ by adjoining a new sort P together with operators $p : \mathrm{sort}(p) \rightarrow P$ for each $p \in P$: The new signature is denoted by Σ, P and a marking is then an element of $T^{\otimes}_{\Sigma,P}(X)$. The input an output functions can also be redefined as $\iota, o : T \rightarrow T^{\otimes}_{\Sigma,P}(X)$, so that for example in figure 2.1 the value of $\iota(t_1)$ is $p_1(x) \otimes p_2(l(x)) \otimes p_2(r(x))$. The firing rule defines the dynamics of the net through the use of substitutions, so that when for example t_1 fires with mode $\{x \leftarrow ph_1\}$ we use the same substitution for all the terms in $\iota(t_1)$. Consequently we need to model the interaction between the monoidal structure and substitution. We want a categorical model in which for a substitution σ, $\sigma(t' \otimes \ldots \otimes t^n) = \sigma(t') \otimes \ldots \otimes \sigma(t^n)$.

The categorical structure suggested by Corradini and Montanari (1992) for the treatment of substitutions is that of a strict cartesian category.

Definition 3.1.15
A *strict cartesian category* is a category $\underline{\mathrm{C}}$ with a terminal object $\perp \in |\underline{\mathrm{C}}|$, and all binary products, such that

$$\begin{aligned} (a \times b) \times c = a \times (b \times c) \qquad & \text{for all } a, b, c \in |\underline{\mathrm{C}}| \\ \perp \times a = a = a \times \perp \qquad & \text{for all } a \in |\underline{\mathrm{C}}| \; . \end{aligned}$$

The category of strict cartesian categories and strict cartesian functors is denoted by $\underline{\mathsf{SCart}}$. □

The idea of using cartesian categories to model algebraic structures was originally proposed by (Lawvere 1963) for the case of single-sorted algebraic theories and by (Benabou 1968) for the many-sorted case, while Goguen (1988) uses the name *substitution system* for essentially the same structure. Given a signature Σ, we can construct a category $\mathsf{SCC}[\Sigma]$ that has as objects the sorts and tuples of sorts of the signature, and as arrows the projections and the operations of the signature. Substitution is modeled by composition while the projections model variables. For example the term $f(g(x), x)$ corresponds to the composition of arrows

$$s_1 \xrightarrow{\langle \pi_1, \pi_1 \rangle} s_1 \times s_1 \xrightarrow{\langle g, id \rangle} s_2 \times s_1 \xrightarrow{f} s,$$

where the pair $\langle \pi_1, \pi_1 \rangle$ represents the fact that we will use the same value x as argument to both g, and f. The idea of using projections for variables has also been proposed by Asperti and Martini (1989) and Corradini and Asperti (1993).

Corradini and Montanari (1992) give an alternative formulation of this construction. It can be formalized by viewing a signature as an element of $\underline{\mathsf{MGraph}} = \underline{\mathsf{Cat}}(\underline{\mathsf{Set}}, \underline{\mathsf{Mon}})$ of graphs with a monoidal structure on the nodes, so that each signature defines a monoidal graph, where each operator $\sigma : s_1 \ldots s_n \to s$ is an arrow. The point is that there exists an adjunction between the categories $\underline{\mathsf{SCart}}$ and $\underline{\mathsf{MGraph}}$. The forgetful functor just forgets about the projections in $\underline{\mathsf{SCart}}$, and the free functor freely adds them and completes the graph into a category. Thus given a signature $\langle \Sigma, P \rangle$ we can generate a strict cartesian category $\mathsf{SCC}[\Sigma, P]$.

Example 3.1.16
Place p_2 in figure 2.1 is of sort $fork$. In $\mathsf{SCC}[\Sigma, P]$ it is represented as an arrow $fork \xrightarrow{p_2} P$. The term $l(x)$ corresponds to an arrow $phil \xrightarrow{l(x)} fork$. The arrows can now be composed to obtain $phil \xrightarrow{p_2(l(x))} P$, that represents $\langle p_2, l(x) \rangle$. □

So the set of places and the specification in an Algebraic net generate a strict cartesian category. However we have for now completely forgotten about the monoidal structure of the places. It turns out that we can describe this structure through the notion of a monoid in the category of strict cartesian categories $\underline{\mathsf{SCart}}$. Thus, as is already hinted at by the expression $T^{\otimes}_{\Sigma,P}(X)$, the monoidal structure is added "on top" of the substitutions. So our places live in $\mathsf{Mod}(\mathcal{SK}_{\underline{\mathsf{SCart}}} \otimes \mathcal{SK}_{\underline{\mathsf{CMon}}}) \simeq \underline{\mathsf{CMon}}(\underline{\mathsf{SCart}})$. Since we obviously have a left adjoint $F : \underline{\mathsf{SCart}} \to \underline{\mathsf{CMon}}(\underline{\mathsf{SCart}})$, we will denote the monoidal substitution system generated by the signature $\mathsf{SCC}[\Sigma, P]$ by $\mathsf{SCC}[\Sigma, P]^{\otimes}$.Thus from the example in figure 2.1 the term

$$\iota(t) = \langle p_1, x \rangle \otimes \langle p_2, l(x) \rangle \otimes \langle p_2, r(x) \rangle$$

would translate into the arrow

$$phil^{\otimes} \xrightarrow{p_1(x) \otimes p_2(l(x)) \otimes p_2(r(x))} P^{\otimes} \ .$$

Suppose now that we wanted to substitute ph_1 for x. This implies that we need to compose $\perp \xrightarrow{ph_1} phil$ with the arrow above. The problem is that the codomain and the domain do not match: the arrow $\perp \xrightarrow{ph_1} phil$ is an arrow in $\mathsf{SCC}[\Sigma, P]$. It can however be lifted to and arrow $\perp^{\otimes} \xrightarrow{ph_1} phil^{\otimes}$. The point here is, that the arrow $\perp \xrightarrow{ph_1} phil$ is the arrow that maps the only element of the terminal set $\perp$ to the element ph_1 of $phil$. The arrow $\perp^{\otimes} \xrightarrow{ph_1} phil^{\otimes}$ is the free extension of this arrow. Thus when composing with the arrow $phil^{\otimes} \xrightarrow{p_1(x)\otimes p_2(l(x))\otimes p_2(r(x))} P^{\otimes}$ we obtain the wanted composition

$$\perp^{\otimes} \xrightarrow{p_1(ph_1)\otimes p_2(l(ph_1))\otimes p_2(r(ph_1))} P^{\otimes} .$$

We have now established the structure of the states. What remains to do is to establish the structure of the transitions and the adjunction between states and transitions. We can think of a transition in an Algebraic net as an operator of sort T, that has as arity the sort string of its variables $Var(t)$. Each net then defines a graph C_{AN} in $\underline{\mathsf{MGraph}}$ with the transitions as operators and the strings of sorts as nodes. For example the philosophers net defines the signature $t_1 : phil \to T, t_2 : phil \to T$. The adjunction between the states and transitions is then obtained as the composition of the adjunctions $\underline{\mathsf{MGraph}} \underset{U}{\overset{F}{\rightleftarrows}} \underline{\mathsf{SCart}} \underset{U}{\overset{F}{\rightleftarrows}} \underline{\mathsf{CMon}}(\underline{\mathsf{SCart}})$.

The graph of an Algebraic net is now a graph, where both the places and the transitions have a natural graph structure associated with them. The transitions are arrows in $\underline{\mathsf{MGraph}}$, while the places are arrows in $\underline{\mathsf{CMon}}(\underline{\mathsf{SCart}})$.

Definition 3.1.17
Given an Algebraic net specification $AN = \langle S, \Sigma, EQ, X, T, P, \iota, o\rangle$ its representation as a heterogeneous graph is:

$$G_{AN} = \langle \mathsf{SCC}[\Sigma, P]^{\otimes}, C_{AN}, \delta_0, \delta_1\rangle,$$

where $\mathsf{SCC}[\Sigma, P]^{\otimes} \in |\underline{\mathsf{CMon}}(\underline{\mathsf{SCart}})|$, and $C_{AN} \in |\underline{\mathsf{MGraph}}|$. The graph morphisms δ_0, δ_1 from C_{AN} to the underlying graph of $\mathsf{SCC}[\Sigma, P]^{\otimes}$ are monoid morphisms on the nodes. Thus δ_0 maps a transition $t : s_1 \ldots s_n \to T$ to an arrow $s_1^{\otimes} \ldots s_n^{\otimes} \to P^{\otimes}$. Analogously for δ_1. □

Before giving an example let us briefly reflect on where we stand in terms of the steps explicated on page 41. We have now established the structure on the states and transitions. The category of states is the category of commutative monoids in a small strict cartesian category, while the category of the transitions is the category of monoidal graphs. The free functor that connects these two is the functor that first completes the monoidal graph with projections making the monoidal operation a product, and then adds a free monoid structure on top of this.

The graph on graph structure is best described by an example.

Example 3.1.18
The graph of the philosopher net is given by the diagram:

$$\begin{tikzcd} phil \arrow[r, "t_1"] & T \\ phil^{\otimes} \arrow[r, "p_1(x) \otimes p_2(l(x)) \otimes p_2(r(x))"] \arrow[dr, "p_3(x)"] & P^{\otimes} \\ & P^{\otimes} \end{tikzcd}$$

$$\begin{tikzcd} phil \arrow[r, "t_2"] & T \\ phil^{\otimes} \arrow[r, "p_3(x)"] \arrow[dr, "p_1(x) \otimes p_2(l(x)) \otimes p_2(r(x))"] & P^{\otimes} \\ & P^{\otimes} . \end{tikzcd}$$

The diagram expresses the fact that

$$\begin{aligned}
\delta_0(t_1 : phil \to T) &= p_1(x) \otimes p_2(l(x)) \otimes p_2(r(x)) : phil^{\otimes} \to P^{\otimes} \\
\delta_1(t_1 : phil \to T) &= p_3(x) : phil^{\otimes} \to P^{\otimes} \\
\delta_0(t_2 : phil \to T) &= p_3(x) : phil^{\otimes} \to P^{\otimes} \\
\delta_1(t_2 : phil \to T) &= p_1(x) \otimes p_2(l(x)) \otimes p_2(r(x)) : phil^{\otimes} \to P^{\otimes},
\end{aligned}$$

that is we have simply just added type information to the terms. □

From the example it is easy to see how the reflexive monoidal structure on the transitions added in step 2 looks like. In $TS(G_{AN}) \in \underline{\mathsf{RGraph}}(\underline{\mathsf{CMon}}(\underline{\mathsf{SCart}}))$, we would for example have an arrow:

$$\begin{aligned}
&\delta_0(t_1 \otimes t_1 : phil^{\otimes} \to T^{\otimes}) = \\
&\qquad p_1(x) \otimes p_2(l(x)) \otimes p_2(r(x)) \otimes p_1(x) \otimes p_2(l(x)) \otimes p_2(r(x)) : phil^{\otimes} \to P^{\otimes} \\
&\delta_1(t_1 \otimes t_1 : phil^{\otimes} \to T^{\otimes}) = \\
&\qquad p_3(x) \otimes p_3(x) : phil^{\otimes} \to P^{\otimes}
\end{aligned}$$

and all other multiples of it. This step also adds idle transitions like

$$\begin{aligned}
\delta_0(t_{f_1} : fork^\otimes \to T^\otimes) &= f_1 : \perp^\otimes \to fork^\otimes \\
\delta_1(t_{f_1} : fork^\otimes \to T^\otimes) &= f_1 : \perp^\otimes \to fork^\otimes
\end{aligned}$$

for each arrow in $\mathsf{SCC}[\Sigma, P]^\otimes$.

The third step involves the dynamics of the net. That is we want to compute the free composition of the arcs in $TS(G_{AN})$ to obtain $[\![AN]\!]_F$ the free model of the transition system.

Suppose that we are at the initial marking of the net in figure 2.1,

$$\perp^\otimes \xrightarrow{p_1(ph_1)\otimes p_1(ph_2)\otimes p_1(ph_3)\otimes p_2(f_1)\otimes p_2(f_2)\otimes p_2(f_3)} P^\otimes,$$

which actually is represented by the corresponding idle transition,

$$\begin{aligned}
&\delta_0(t_{p_1(ph_1)\otimes p_1(ph_2)\otimes p_1(ph_3)\otimes p_2(f_1)\otimes p_2(f_2)\otimes p_2(f_3)} : \perp^\otimes \to T^\otimes) = \\
&\qquad p_1(ph_1) \otimes p_1(ph_2) \otimes p_1(ph_3) \otimes p_2(f_1) \otimes p_2(f_2) \otimes p_2(f_3) : \perp^\otimes \to P^\otimes \\
&\delta_1(t_{p_1(ph_1)\otimes p_1(ph_2)\otimes p_1(ph_3)\otimes p_2(f_1)\otimes p_2(f_2)\otimes p_2(f_3)} : \perp^\otimes \to T^\otimes) = \\
&\qquad p_1(ph_1) \otimes p_1(ph_2) \otimes p_1(ph_3) \otimes p_2(f_1) \otimes p_2(f_2) \otimes p_2(f_3) : \perp^\otimes \to P^\otimes
\end{aligned}$$

and recall from previously, that we can compose $\perp^\otimes \xrightarrow{ph_1} phil^\otimes$ with $phil^\otimes \xrightarrow{p_1(x)\otimes p_2(l(x))\otimes p_2(r(x))} P^\otimes$ to obtain $phil^\otimes \xrightarrow{p_1(ph_1)\otimes p_2(l(ph_1))\otimes p_2(r(ph_1))} P^\otimes$, and analogously for $p_3(x)$. We can fire transition t_1 by substituting ph_1 for x, ie. composing $ph_1 : 1 \to phil$ with $t_1 : phil \to T$, giving the transition instance $1 \xrightarrow{ph_1;t_1} T$. The instance is then

$$\perp^\otimes \xrightarrow{\quad ph_1;t_1 \quad} T^\otimes$$

$$\begin{tikzcd}
\perp^\otimes \arrow[r, "p_1(ph_1)\otimes p_2(l(ph_1))\otimes p_2(r(ph_1))"] \arrow[dr, "p_3(ph_1)"] & P^\otimes \\
& P^\otimes
\end{tikzcd}$$

Then we match the lhs to the initial marking and substitute the lhs with the rhs in the initial marking obtaining the transition:

$$\begin{tikzcd}
\perp^{\otimes} \arrow[rr, "(ph_1;t_1)\otimes t_{p_1(ph_2)\otimes p_1(ph_3)\otimes p_2(f_3)}"] & & T^{\otimes} \\
\perp^{\otimes} \arrow[rr, "p_1(ph_1)\otimes p_1(ph_2)\otimes p_1(ph_3)\otimes p_2(f_1)\otimes p_2(f_2)\otimes p_2(f_3)"] \arrow[drr, "p_1(ph_2)\otimes p_1(ph_3)\otimes p_2(f_3)"] & & P^{\otimes} \\
 & & P^{\otimes}
\end{tikzcd}$$

Naturally if we had enough forks we could fire t_1 in parallel with two different forks obtaining

$$ph_1;t_1 \otimes ph_2;t_1 \ .$$

In this way we obtain all the required compositions. The free model is thus $[\![AN]\!]_F = \mathsf{C}_{\underline{\mathsf{CMon}}(\underline{\mathsf{SCart}})}[\mathsf{TS}[G_{AN}]]$.

Naturally the model $[\![AN]\!]_F$ contains many more arrows than the transition system obtained in section 3.1.1. For example in the philosopher example we have an arrow

$$t_1;t_2 : p_1(x)\otimes p_2(l(x))\otimes p_2(r(x)) \rightarrow p_1(x)\otimes p_2(l(x))\otimes p_2(r(x)) \ .$$

This arrow gives us a kind of abstract behavior.

We can however recover the semantics of section 3.1.1.

Proposition 3.1.19
Let AN be an Algebraic net and $[\![AN]\!]_F$ its free model. Then the set of all arrows of $\mathsf{SCC}[\Sigma, P]^{\otimes}$ of the form $\perp^{\otimes} \rightarrow P^{\otimes}$ reachable in $[\![AN]\!]_F$ from the arrow $!^{\otimes} : \perp^{\otimes} \rightarrow \perp^{\otimes}$ is isomorphic to the set of reachable states of AN.

Proof:
Follows from the fact that the set of arrows from 1 in $\mathsf{SCC}[\Sigma, P]$ is isomorphic to $P \times T_{\Sigma/\equiv}$. □

The semantics for Petri nets $\underline{\mathsf{CatPetri}}$ is recovered in the case when the Algebraic Net is essentially a Petri net.

Proposition 3.1.20
Let AN_T be an Algebraic net with an empty specification. Then $\perp^{\otimes} \downarrow [\![AN_T]\!]_F$, the set of arrows from $\perp^{\otimes}$ in $[\![AN_T]\!]_F$ is isomorphic to $\underline{\mathsf{CatPetri}}(\mathsf{U}(AN_T))$.

Proof:
The empty algebraic specification generates an empty a substitution system.

Thus the substitution system $\mathsf{SCC}[\Sigma, P]$ only contains the constants $p : \bot \to P$ for all $p \in P$ and all the tuples. Then the set of arrows $\bot^{\otimes} \to P^{\otimes}$ in $\mathsf{SCC}[\Sigma, P]^{\otimes}$ is the same as $P^{\otimes}$. In the same way, the monoidal graph generated by the set of transitions consists of the arrows $t : 1 \to T$ for each $t \in T$, which means that the graphs is essentially the set T. If one now goes through the construction it is easy to see that the proposition follows. $\square$

We can now complete the method of Corradini and Montanari by selecting the intended models of the transition system. These are those objects $\underline{\mathsf{C}}$ of $\underline{\mathsf{Cat}}(\underline{\mathsf{CMon}}(\underline{\mathsf{SCart}}))$ such that there exists an internal functor $F : [\![AN]\!]_F \to \underline{\mathsf{C}}$ that is an isomorphism on objects and an epimorphism on arrows. The objects of $\underline{\mathsf{C}}$ differ from those in $[\![AN]\!]_F$ only by having certain arrows identified. These objects correspond to the different interpretations of an Algebraic net specification.

In this section we have identified high-level nets as graphs on a monoidal substitution system, and as a byproduct we have identified the notion of substitution and commutative free monoid as the notions setting up the dynamics of the net. It is interesting to note, that Husberg (1992) arrives to a similar definition by directly generalizing the notion of an operator in an algebraic theory, where an algebraic theory is essentially a substitution system.

Our model of Algebraic nets is general in the sense that by changing different components of the construction we obtain new high-level formalisms. Indeed if one analyses the techniques used in this section, a specification formalism construction toolkit suggests itself. The components are sketches, the tensor product of sketches, and adjoints between the corresponding models. In this section the sketches were the sketch of commutative monoids, the sketch of small strict cartesian categories, the sketch of graphs and the sketch of categories, ie. the transition system is model of $\mathcal{SK}_{\underline{\mathsf{CMon}}} \otimes \mathcal{SK}_{\underline{\mathsf{SCart}}} \otimes \mathcal{SK}_{\underline{\mathsf{Graph}}}$ while the behavior is a model in $\mathcal{SK}_{\underline{\mathsf{CMon}}} \otimes \mathcal{SK}_{\underline{\mathsf{SCart}}} \otimes \mathcal{SK}_{\underline{\mathsf{Cat}}}$. The "driving force" behind this construction is the adjunction between graphs and categories. This adjunction lifts through the tensor product to give us the free model of the transition system. Then by changing any of these components we would obtain some other "high-level" formalism. For example we could select a different substitution system: Goguen (1988) list a number of interesting substitution systems, among them order-sorted algebra. This gives us formalisms that are still net like. A more radical idea is to change the sketch of the monoids to something else. One possibility would be to axiomatize some process algebra as a sketch. This would give us the possibility to extend the treatment of process algebras of (Ferrari 1990) to process algebras with value passing.

3.2 A sheaf semantics

Sheaf theory is a mathematical tool that has been successfully applied to the solution of difficult mathematical problems. Gray (1979) surveys applications to complex analysis, algebraic geometry, differential equations, and category theory. Sheaf theory has also been used in a general formulation of systems theory by Goguen in (Goguen 1973, Goguen 1975, Goguen and Ginali 1978). The basic building blocks of this categorical formulation of General Systems Theory besides sheaves are the categorical notions of diagram, limit and co-limit.

In the paradigm the behavior of a component, or subsystem, is represented by a sheaf. The sheaf describes those *observations* that can be made about the behavior of the component. The interconnection of components to a system is described by a diagram of sheaves. The behavior of the system is then calculated by taking the limit of its diagram. The limit consists of a product of those observations of component behavior that are mutually consistent. Finally co-limits are used to connect different systems together to form larger modules. This paradigm was later extended to models of concurrency (Goguen 1992) and used to give a semantics for the FOOPS language (Wolfram and Goguen 1991).

In this section we apply the "Objects are sheaves" paradigm as formulated by Goguen (1992) to one specific model of concurrency, namely Petri nets. The main result of this section is that both interleaving, as well as a non-interleaving semantics for Petri nets arise through the categorical process of taking the limit.

We will start with a very simple view of a transition in a Place/Transition net. This view will then be refined giving us an interleaving semantics with a global firing rule. Then we will focus our attention on Elementary net systems. We will further refine our semantics so that we will obtain a non-interleaving semantics for Elementary net systems. The results presented in this section have been reported in (Lilius 1993).

3.2.1 Sheaves and transitions

The basic idea in the "Objects are sheaves" paradigm is that an object has an internal state that can change, and that this state can only be observed through slots that are called attributes. From an external point of view, an object thus consists of a set of attributes, and the only thing we know about the object is that what we can observe about its attributes. It is actually these observations that will be modeled as sheaves. Observations may be partial in the sense that at the current moment we have only observed certain parts of the object, but the observations should be consistent in the sense that partial or local observations can be "pasted" together to global observations.

The observations will be formalized in terms of functions $f : U \to A$ where U is some space-time domain and A is the domain of the attributes of the object, so that $f(u)$ tells us what can be observed about the object from point u. The elements of the domain U are partially ordered by inclusion, and closed under finite intersection and arbitrary union, ie. form a topological space, but in this section the following definition is sufficient:

Definition 3.2.1
A *base* $\underline{T}$ for observations is a family of sets partially ordered by inclusion.
□

Typically we will use *discrete linear time* as our base:

$$I_0(\omega) = \{\varnothing, \{0\}, \{0,1\}, \{0,1,2\}, \ldots\} \cup \{\omega\} \, .$$

An object O then consists of observations over a base $\underline{T}$. The consistency requirement can now be formulated as follows. Given an inclusion $U \subseteq V$ and an observation $f : V \to A$ we can form the restriction of f to U and we will denote it by $f \restriction U$. Let $O(U)$ denote the observations over U. Given the inclusion $i : U \rightarrowtail V$, we can form the map $O(i) : O(V) \to O(U)$ that maps an observation f over V to its restriction $f \restriction U$. O now satisfies the following two equations $O(i;j) = O(j);O(i)$, $O(1_U) = 1_{O(U)}$, that make it into a contravariant functor.

Definition 3.2.2
A *presheaf* is a functor $O : \underline{T}^{op} \to \underline{C}$ where $\underline{T}$ is the *base category* and $\underline{C}$ is the *structure category*. If $i : U \rightarrowtail V$ is in $\underline{T}$, then $O(i) : O(V) \to O(U)$ is called the *restriction* morphism induced by i. □

The idea that observations are closed under restriction can be seen as a categorical formulation of the *prefix closure* property of traces.

The structure category is the domain of the attributes. By chosing a suitable category many different examples can be modeled, as described in (Goguen 1992). In this section the attributes will have a very simple form, which allows us to give the definition of a preobject in a simple form.

Definition 3.2.3
A *preobject* O over a base $\underline{T}$ with *attribute object* $A = A_1 \times \cdots \times A_n$ is a presheaf of the form

$$O(U) = \{h : U \to A_1 \times \cdots \times A_n \mid K(h)\}$$

where the morphisms $O(i)$ are restriction maps, and the relation K expresses some property of the functions, embodying the "laws" that O satisfies; the elements of A may be called states. □

Example 3.2.4 ((Goguen 1992))
The behavior of a capacitor is governed by a linear differential equation. Let $\mathcal{I}_0(\mathcal{R}^+) = \{[0, r) | r \geq 0\} \bigcup \{\mathcal{R}^+\}$ be the base consisting of semi-open intervals of real numbers starting at time 0. Then the behavior of the capacitor is described, for $I \in \mathcal{I}_0(\mathcal{R}^+)$, by the sheaf

$$OI = \{f : I \to \mathcal{R}^3 \;\mid\; c = F\frac{d}{dt}(u_i - u_o), \;\; f \text{ is } C^\infty \text{ on } I\},$$

where F is the capacitance, c is the current through the capacitor and u_i and u_o are the voltages at the input and output of the capacitor. In this case the structure category could be specified more exactly eg. as the category of Banach spaces.

The observations are thus sequences of triples (c, u_i, u_o) that form a (partial) solution to the differential equation. □

The definition of an object is the following (Goguen 1992).

Definition 3.2.5
A preobject O is called an *object* if its base $\underline{\mathrm{I}}$ is a topological space, and if given $U_i \in \underline{\mathrm{I}}$ and $f_i \in O(U_i)$ for all $i \in I$ such that $U = \bigcup_{i \in I} U_i$ and $f_i \restriction U_i \cap U_j = f_j \restriction U_i \cap U_j$ for all $i, j \in I$, then there exists a unique $f \in O(U)$ such that $f \restriction U_i = f_i$ for all $i \in I$. This condition is called the *sheaf condition.* If the index set I is restricted to be finite, then the corresponding condition is called *finite sheaf condition.* □

Informally, this means that if observations are consistent over a subinterval, then these observations can be pasted together into an observation that contains them all as sub-observations. Depending on whether one compares a pair or a possibly infinite set of observations, one speaks of a finite and an infinite sheaf condition respectively.

The basic idea in the categorical formulation of General Systems Theory (Goguen 1973, Goguen 1975, Goguen and Ginali 1978) is that the behavior of a component is represented by a sheaf. The whole system is then represented by a diagram of these sheaves where the diagram tells how the different components are interconnected. Now because a transition is a component of a net, a transition will be modeled by a sheaf. In the next section we will show how a net corresponds to a diagram of sheaves and the behavior of the net to the limit of the diagram.

Before we go into details, we need some definitions from net theory and some notation. Contrary to the previous sections and the forthcoming sections, we will not use the categorical definition of a Petri net. The reason is that we don't need the free construction. Instead we use a slightly massaged definition of a Place/Transition net without capacities (Reisig 1986).

Definition 3.2.6
A *Place/Transition net* N is a quadruple $\langle T, P, \iota, o\rangle$ where T is the set of transitions, P is the set of places, and $\iota, o : T \to [P \to N]$ are the input and output weight functions and N is the set of natural numbers. □

It is customary to assume that the functions ι and o are injective so as to rule out transitions with identical sets of input and output places. Such nets are called "simple" or in the terminology of (Corradini 1990) 'extensional transition systems'.

The following notations will be useful, where $t \in T$ and $p \in P$. The *preset* or set of *input places* of a transition t is defined as ${}^\bullet t = \{p \mid \iota(t)(p) \geq 1\}$. The *postset* or the set of *output places* of a transition t is defined as $t^\bullet = \{p \mid o(t)(p) \geq 1\}$. Finally the restrictions of the input and output weight functions ι, o to the pre- and postsets are defined as $\iota_t = \iota(t) \restriction {}^\bullet t$ and $o_t = o(t) \restriction t^\bullet$ respectively.

The intuition behind P/T nets is that a place can be occupied by tokens. The distribution of tokens in the net represents the state of the system and it is called a *marking*. If the number of tokens in the input places of a transition exceeds the input weight of that transition, then the system can change state by *firing* that transition. This process is formally defined through the firing rule:

Definition 3.2.7
Given a Petri net $N = \langle T, P, \iota, o\rangle$, a *marking* is a function $M : P \to N$ mapping places to integers. In a net N we say that a transition $t \in T$ is *enabled at M* iff $\iota(t) \leq M$. If a transition is enabled it can fire, producing a new marking through the *firing rule*:

$$M' = M - \iota(t) + o(t),$$

where the addition and subtraction of integers is extended to addition and subtraction of integer valued functions by defining $(f + g)(x) = f(x) + g(x)$ and $(f - g)(x) = f(x) - g(x)$. □

Armed with these definitions we can give the following definition of the sheaf corresponding to a transition:

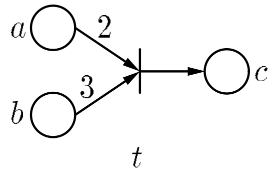


Figure 3.2: A transition t.

Definition 3.2.8
Given a transition t in a P/T net we define a sheaf as follows:

$$t(I) = \left\{ f : I \rightarrow N^{|{}^\bullet(t)|+|({}^\bullet t)|} \mid f(i+1) = f(i) - \iota_t + o_t \quad \text{if } \iota_t \leq f(i) \right\},$$

with $i, i+1 \in I$. □

The definition describes those observations that arise from the firing of a transition. The idea is that we only look at the input and output places of the transition. What we observe are integers that represent the number of tokens in the places. Thus the attributes of the transition are integers. The specific values of the attributes are not specified as the environment may add or remove tokens. However, whenever the firing condition is met, the transition must fire. In that case the observation $f(i+1)$ is related to the the observation $f(i)$ by the firing equation.

Example 3.2.9
Take the transition in figure 3.2. The corresponding sheaf is:

$$\begin{aligned} t = \{ & f : I \rightarrow N \times N \times N \mid \\ & f(i+1) = f(i) - (2,3,0) + (0,0,1) \;\; \text{if } (2,3,0) \leq f(i) \} \end{aligned}$$

□

One possible sequence of observations is

i	0	1	2	3	4
f	$(0,0,0)$	$(1,0,1)$	$(2,3,0)$	$(0,0,1)$	$(0,0,1)$

where the transition fires at $i = 2$. In the sequence of observations the steps from $i = 0$ to $i = 1$ and $i = 1$ to $i = 2$ are caused by the places becoming marked through the environment. Thus the sheaf is a local constraint on the observations.

A more interesting example is given by nets with time. There are several different ways for incorporating time into nets. We have chosen to attach a firing time to each transition that expresses the number of clock ticks it takes for the transition to fire. This is a very simple model of nets with time. A more elaborate model would attach a time interval to the transition that says within which time span the transition will fire after it has become enabled.

To model the duration of the firing we add a stream that stores the firing time of the transitions:

Definition 3.2.10
Given a transition t in a P/T net with firing duration τ_t we can define a sheaf $t(I)$ as $f : I \to N^n \times N$ given by

$$f(i+1) = \begin{cases} \langle f_{\{1,n\}}(i) - \iota(t), \tau_t \rangle & \text{if } \iota(t) \leq f_{\{1,n\}}(i) \text{ and } f_{n+1}(i) = 0 \\ \langle f_{\{1,n\}}(i), f_{n+1}(i) - 1 \rangle & \text{if } f_{n+1}(i) > 1 \\ \langle f_{\{1,n\}}(i) + o_t, 0 \rangle & \text{if } f_{n+1}(i) = 1, \end{cases}$$

where $f_{\{1,n\}}(i)$ are the n first components of $f(i)$ and $f_{n+1}(i)$ is the $n+1$:st (we abbreviate $n = |^{\bullet}(t)| + |(^{\bullet}t)|$). □

The idea behind this construction is that the $n+1$-st component stores the remaining time before the firing ends as a natural number. The first clause of the case expression corresponds to the beginning of the firing of the transition. The tokens are removed from the input places and the timer is set. The second clause takes care of the countdown by decreasing the counter. After the firing time has expired the third clause outputs the required tokens to the output places and resets the timer to 0. Transitions for other kinds of nets can be treated in an analogous manner.

3.2.2 Nets, systems and their behavior

The intuition underlying this section is that the interconnection of objects should be achieved through morphisms. The system is represented as a diagram and the composite behavior of the system will arise as the limit of the diagram. The process of taking the limit selects mutually consistent behaviors of the subsystems as the behavior of the whole system. In this section we will show how the behavior of a Petri net arises as the limit of its corresponding sheaf diagram.

We start by defining a notion of morphism for pre-objects and objects:

Definition 3.2.11
Given preobjects O and O' over the same base $\underline{T}$, a *morphism* $\phi : O \to O'$ is a family $\phi_U : O(U) \to O'(U)$ of maps, one for each $U \in \underline{T}$, such that for each

$i : U \to V$ in $\underline{\mathsf{T}}$ the diagram

$$
\begin{tikzcd}
O(V) \arrow[r, "\phi_V"] \arrow[d, "O(i)"'] & O'(V) \arrow[d, "O'(i)"] \\
O(U) \arrow[r, "\phi_U"'] & O'(U)
\end{tikzcd}
$$

commutes in $\underline{\mathsf{C}}$. When O and O' are objects, we may also call ϕ an *object morphism* or *sheaf morphism*. This gives rise to categories $\underline{\mathsf{PreObj}}(\underline{\mathsf{T}}, \underline{\mathsf{C}})$, and $\underline{\mathsf{Obj}}(\underline{\mathsf{T}}, \underline{\mathsf{C}})$ of preobjects and objects respectively, over a given base $\underline{\mathsf{T}}$ and structure category $\underline{\mathsf{C}}$ □

A system consists of a diagram of pre-objects or objects.

Definition 3.2.12
A *system* $\mathcal{S}$ consists of a graph with nodes $n \in \mathbf{N}$ labeled by (pre-) objects S_n and with edges $e : n \to n'$ labeled by morphisms $\phi_e : S_n \to S'_n$. □

Morphisms can be used to represent sharing and inheritance of attributes between objects. In the sequel we shall mostly be interested in interconnecting objects through special morphisms called projection morphisms.

Definition 3.2.13
Given preobjects O and O' with attribute objects $A = \Pi_{j \in J} A_j$ and $A' = \Pi_{j \in J'} A_j$ respectively with $J' \subseteq J$, if $a : A \to A'$ sending $\langle a_j | j \in J \rangle$ to $\langle a_j | j \in J' \rangle$ induces a morphism, then it is called a *projection morphism.*

□

A projection morphism "picks" the necessary information from the attribute object. A typical use for projection morphisms is in connection with so called *event streams* $E = \{f \to A | M(f)\}$. An event stream contains the observations pertaining to an attribute. Examples of event streams include wires, communication channels, and places in Petri nets.

To construct a net from a set of transitions, some way of expressing that two transitions are connected to the same place is needed. This is where event streams and projection morphisms come in handy. A place is represented by an event stream $P = \{f : I \to N\}$. If there is a capacity k attached to the place, then the event stream is of the form $\{f : I \to N | f(i) \leq k \;\; i \in I\}$. Each transition is connected to its input and output event streams by projection morphisms. The righthand side of figure 3.3 depicts the diagram obtained from the net on the lefthand side.

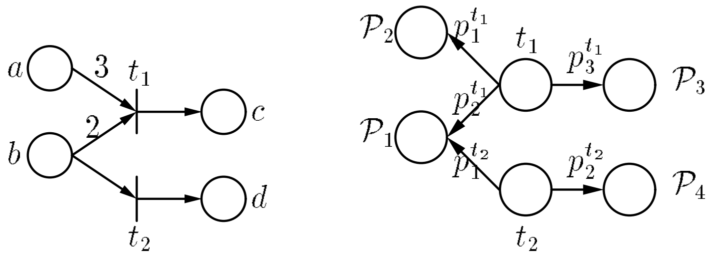


Figure 3.3: The net of the example and its corresponding diagram.

Given a system as a diagram, we can construct its behavior as the limit of the diagram. This limit will contain all the mutually consistent behaviors of the system. The limit can be understood as a completion of the diagram with an object that makes the whole diagram commute.

Definition 3.2.14
Given a diagram whose nodes are labeled by sheaves S_n for $n \in \mathbf{N}$, the behavior object L of the diagram has for each $U \in L$:

$$L(U) = \{\{f_n | n \in \mathbf{N}\} | f_n \in S_n \wedge \phi_e : S_n \to S'_n \Rightarrow \phi_e(f_n) = f'_n\}$$

□

The limit object of the diagram in figure 3.3 is of the form

$$L = \{I \to N^9\} .$$

There will be projections $p_i : L \to P_i$ for $i \in \{1, 2, 3, 4\}$ and projection tuples $\langle p_5, p_6, p_7 \rangle : L \to t_1$ and $\langle p_8, p_9 \rangle : L \to t_2$. The limit object is isomorphic to the object $L' : I \to N^4$ which is the state space of the system.

Actually, the approach presented above is too simplistic, because it does not take into account the global nature of the notion of state. To see this consider the net in figure 3.3. For simplicity all the arcs have a weight of 1, so that the net is essentially an EN-system. Given the marking $M = \{a \mapsto 1, b \mapsto 1, c \mapsto 0, d \mapsto 0\}$, there is a choice or conflict between the firing of t_1, giving the marking $M' = \{a \mapsto 0, b \mapsto 0, c \mapsto 1, d \mapsto 0\}$, and the firing of t_2, resulting in the marking $\{a \mapsto 0, b \mapsto 0, c \mapsto 0, d \mapsto 1\}$. So from the marking $\{a \mapsto 1, b \mapsto 1, c \mapsto 0, d \mapsto 0\}$ it is impossible to get to the marking $\{a \mapsto 0, b \mapsto 0, c \mapsto 1, d \mapsto 1\}$. But if we take the limit of the corresponding sheaf diagram, the limit object will contain this marking, because we have no way of specifying the fact that firing t_1 and t_2 at the same time is an inconsistent behavior. Consequently the integrity of the tokens is destroyed.

To see how to approach a solution to this problem, let us first briefly look at the solution adopted in net theory. The general assumption there is that a system can resolve a conflict by itself. A consequence of this is that a semantics for Petri nets must define an execution policy is some way, explicitly or implicitly. The firing rule for P/T nets establishes such a policy. It says that one should look at one transition at a time to see if it is enabled. During this process of examination each enabled transition is fired. The resulting reachability graph gives an interleaving semantics for the system, where at each instant only one transition is allowed to fire. What the firing rule does not say is in what order the transitions are to be examined.

The net effect of this is that we have a global state and we are able to observe the effect of a transition on the global state. As defined by definition 3.2.8, the firing rule is "distributed" into each sheaf, and thus gains a local character resulting in the unfaithful rendering of the semantics of choice.

To rectify this problem we need to introduce a global scheduler. This sheaf will be connected to each transition of the system and allows only one transition to fire at each instant.

We use the shorthand $T_\perp$ for $T \cup \{\perp\}$.

Definition 3.2.15
Given a set of transitions T define a scheduler sheaf by:

$$T(I) = \{I \to T_\perp\} \ .$$

The possible observations are constrained by the rules:

$$\begin{array}{rcl} t & \mapsto & \perp \text{ with } t \in T \\ \perp & \mapsto & t \text{ with } t \in T \end{array}$$

where the lhs is the observation $f(i)$ and the rhs is the observation $f(i+1)$.

□

The sheaf T is used to store the name of the transition that is currently firing, and the token $\perp$ is used to denote the fact that no transition is in the process of firing. The last rule is applicable when a transition starts to fire. Note that if we would add a rule $\perp \mapsto \perp$ that expresses the fact that it is not necessary for any transition to fire, the system could idle. In such a case the system would not necessarily satisfy a progress assumption.

The definition of a transition sheaf as given in definition 3.2.8 has to be modified to accommodate the scheduler.

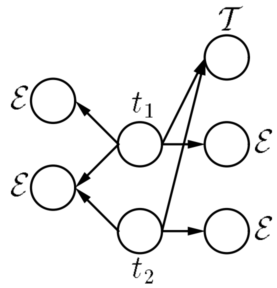


Figure 3.4: The diagram corresponding to the net in figure 3.3.

Definition 3.2.16
Given a transition t in a P/T net we define a sheaf $t(I)$ as

$$f : I \to N^{|^{\bullet}(t)|+|(^{\bullet}t)|} \times T_{\perp}$$

given by

$$f(i+1) = \begin{cases} \langle f_{\{1,n\}}(i), t\rangle & \text{if } \iota_t \leq f_{\{1,n\}}(i) \text{ and } f_{n+1}(i) = \perp \\ \langle f_{\{1,n\}}(i) - \iota_t + o_t, \perp\rangle & \text{if } f_{n+1}(i) = t \end{cases}$$

where $f_{\{1,n\}}$ are the n first components of the $n+1$-tuple $f(i)$. □

The definition sets things up so that firing is only possible if the attribute T has the value $\perp$. The first clause insures that transition will only fire if no other transition is firing, while the second clause does the actual firing. Notice that this definition does not rule out the possibility that the firing process takes an infinite amount of time. This possibility is ruled out by the definition of the sheaf T which has a rule that says that the firing takes only one time tick. Given a net N denote its translation to a diagram of sheaves with scheduler by $[\![N]\!]_{PTSh}$.

Now the net in figure 3.3 is translated to the diagram in figure 3.4. The limit-object of this diagram will have of the form

$$\{f : I \to N \times N \times N \times N \times T_{\perp} \times (N \times N \times N \times T_{\perp}) \times (N \times N \times T_{\perp})\},$$

where the first four N:s correspond to the $\mathcal{E}$:s in figure 3.3 the last two expressions in parenthesis to the transitions and the $T_{\perp}$ to T. This limit-object is isomorphic to the following simpler object

$$\{f : I \to N \times N \times N \times N \times T_{\perp}\},$$

because the arrows in the diagrams represent projections. From the structure of the object it is easy to see that only one transition is able to fire at each instant. The limit object corresponds to what is more commonly called the

i	0	1	2
f	$(0,0,0,0,\perp)$	$(1,1,0,0,t_1)$	$(0,0,1,0,\perp)$
g	$(0,0,0,0,\perp)$	$(1,1,0,0,t_2)$	$(0,0,0,1,\perp)$

Figure 3.5: Some observations of the net in figure 3.3.

state space of the net. From this object one can see the state of the system by projecting on all but the last component.

Some possible observations are given in figure 3.5.

In general we have the following theorem.

Theorem 3.2.17
Let N be a Petri net with initial marking M_0. Let $[\![N]\!]_{PTSh}$ be its corresponding diagram of sheaves. Then the set of paths from the initial marking $M_0 \xrightarrow{t_0} M_1 \xrightarrow{t_1} \ldots$ is isomorphic to the observations of the limit object of $[\![N]\!]_{PTSh}$ with initial observation M_0.

Proof:
($\Rightarrow$) To each path corresponds an observation: for $M_0 \xrightarrow{t_0} M_1$ the observation is $0 \mapsto \perp \times M_0$, $1 \mapsto t \times M_0$, $2 \mapsto \perp \times M_1$. Assume we have a path $M_0 \xrightarrow{t_0} M_1 \ldots M_{i-1} \xrightarrow{t_i} M_i$ for i then for $i+1$ we have $M_i \xrightarrow{t_i} M_{i+1}$ and the observation is $2i \mapsto \perp \times M_i$, $2i+1 \mapsto t_i \times M_i$, $2(i+1) \mapsto \perp \times M_{i+1}$.

($\Leftarrow$) To each observation corresponds a path: Follows from the definition of the sheaf 3.2.16. □

A more interesting semantics can be obtained by coding the arbitration of choice locally. To simplify the presentation we will restrict our nets so that all places have a capacity of one and all arcs have a weight of one, making the nets essentially Elementary net (EN) systems (cf. (Thiagarajan 1986)). When talking about EN-systems, it is customary to call places conditions, and transitions events. This construction will give the non-interleaving semantics mentioned in the introduction.

The following two definitions set up the basic framework where we assume that we are given an elementary net with set of conditions C and set of events E:

Definition 3.2.18
A condition is represented by the sheaf:

$$C(I) = \{f : I \to E \cup \{\mathbf{true}, \mathbf{false}\}\},$$

which is governed by the rules:

$$\begin{array}{rcl} e & \mapsto & m \text{ with } e \in E \text{ and } m \in \{\mathbf{true}, \mathbf{false}\} \\ m & \mapsto & e \text{ with } e \in E \text{ and } m \in \{\mathbf{true}, \mathbf{false}\} \end{array}$$

□

The constants **true** and **false** are used to tell whether the condition holds or not. Now because we do not have a global scheduler the name of the event is used to schedule the execution locally.

Definition 3.2.19
An event is modeled by the sheaf:

$$E(I) = \{f : I \to (e \cup \{\mathbf{true}, \mathbf{false}\})^{|{}^\bullet e| + |e^\bullet|},$$

which is governed by the transition rules:

$$\begin{array}{rcl} \langle x_1, \ldots, x_n \rangle & \mapsto & \langle x'_1, \ldots, x'_n \rangle \text{ with } |\langle x_1, \ldots, x_n \rangle| = |\langle x'_1, \ldots, x'_n \rangle| = n \\ & & \text{and with } e \neq e' \text{ for all } e' \in \{x_1, \ldots, x'_n\} \\ \langle \mathbf{true}^k, \mathbf{false}^l \rangle & \mapsto & \langle e^k, e^l \rangle \\ \langle e^k, e^l \rangle & \mapsto & \langle \mathbf{false}^k, \mathbf{true}^l \rangle \end{array}$$

where $k = |{}^\bullet e|, l = |e^\bullet|$ and $n = k + l$. □

An event is represented by the product of its pre- and post-conditions. An event can only occur when all of its preconditions hold and all of its post-conditions don't (rule 2, lhs). When an event occurs it replaces all pre- and postconditions with its name (rule 2, rhs). This prevents other events that are in conflict from occurring, and implements conflict resolution on a local level. The actual firing is done by rule 3, while rule 1 takes care of the case when some other event is chosen. Again if we would add a rule $\langle m^k, n^l \rangle \mapsto \langle m^k, n^l \rangle$ with $m, n \in \{\mathbf{true}, \mathbf{false}\}$ that expresses the fact that it is not necessary for any event to happen, the system could idle. In such a case the system would not necessarily satisfy a progress assumption. By using event sheaves of the form $\{f : I \to \{\mathbf{true}, \mathbf{false}\}\}$ we can now construct a diagram of a given elementary net and then form the limit of this diagram to obtain the semantics of the system.

Let us finally briefly look at how our semantics reflects conflict and concurrency, to see in which sense we have a non-interleaving semantics. The resolution of choice is distributed into the conditions by rule 2, so that instead of one global scheduler, each place acts as an arbiter for the events connected to it. Thus the resolution of choice is done locally. It is interesting to note that a similar solution was adopted by Bütler, Esser and Mattman (1991) in an implementation of Petri nets on a transputer network. The lack of a global scheduler also means that causally independent transitions may fire in parallel, thus giving a non-interleaving semantics. That is, the limit object will contain observations where causally independent transitions will fire in parallel. Thus the semantics is a non-interleaving semantics.

Chapter 4

Foldings

General Net theory as initiated by C.A. Petri (1973) is the study of nets and their morphisms. A folding is a special kind of morphism between nets, that has been used to relate a net to its causal behaviors, its processes, and to describe and motivate high-level nets (Genrich, Lautenbach and Thiagarajan 1980). Recently the construction of strict high-level nets from Condition/Event systems has been formalized and studied (Smith and Reisig 1987). In this chapter we study the construction of non-strict high-level nets from Petri nets through foldings.

The setting for our study of foldings is the categorical theory of Petri nets proposed by Meseguer and Montanari (1990). In their approach Petri nets (Place/Transition nets) are presented as graphs with a monoidal structure on nodes. Contrary to General Net theory, where a morphism is defined as a function on the union of transitions and places, a morphism here is taken to be a pair of functions between the set of transitions and the monoid of places respectively, such that the underlying morphism is a graph morphism[1]. Such a morphism is called a homomorphism in the context of General Net theory. Also, in the category of Petri nets, a process of a net is not related to the net itself in the same way as in General Net theory. Thus processes defined through foldings cannot be taken as the fundamental definition of causal behavior of a net. In Meseguer and Montanari's approach this problem is addressed by a categorical notion known as a *free construction*. To a given net we can associate a new net, that has as new transitions all possible (parallel) transition sequences of the original net. Different notions of causal behavior are then obtained as suitable equivalence classes of arrows (Degano, Meseguer and Montanari 1989). However the inverse of folding, *unfolding* remains an important notion in the categorical theory of high-level nets (Dimitrovici,

[1] For a short review of this approach see appendix A.2

Hummert and Pétrucci 1991), because it relates a high-level net to a low-level net with in some sense identical behavior.

In this chapter we define a notion of folding for Petri nets as a special kind of morphism in the category of Petri nets. We then show how to a given folding we can associate an Algebraic Net whose underlying net is isomorphic to the codomain of the folding and whose unfolding is isomorphic to the domain of the folding. It is at first not clear if this construction is functorial, but by analyzing the structure of the category of Petri nets we are able to identify a preorder on nets that allows us to prove the functoriality.

The main contributions of this chapter are:

- The formalization of the folding construction of non-strict high-level nets from Petri nets.
- The identification of a preorder on nets in the category of Petri nets.
- A generalization of the folding construction to Algebraic nets.
- The extension of Findlow's "refolding" construction to Algebraic nets.

The chapter is structured as follows. We first define the categories of Petri nets and Algebraic nets. We also define several functors between these. Then we define the notion of folding in the category of Petri nets and show how it gives rise to an Algebraic net. This construction is proved to be functorial. We then show how we can define a notion of folding directly on the Algebraic net. Finally we discuss how the construction can be used to transform a net so that is has a deadlock preserving skeleton. The results presented in this chapter have been presented in (Lilius 1994a) and (Lilius 1994b).

4.1 Foldings

As stated earlier the aim of General Net theory as introduced by C.A. Petri (1973) is to study morphisms of nets. Foldings play a central role in this theory, because they are used to relate different nets and net-classes. In this chapter we are interested in the use of foldings to relate different net-classes. A study of foldings using the classical notion of net-morphism has been done by Smith and Reisig (1987). In this section we will define the notion of folding within the categories $\underline{\mathsf{PetriG}}$ and $\underline{\mathsf{AN}}$.

The material in this section differs from the material presented by Smith and Reisig (1987) in two respects. First, the notion of morphism is different. The classical net-morphism (Petri 1973) allows a map to take transitions to places, while morphisms in $\underline{\mathsf{PetriG}}$ are essentially graph-morphism, and thus respects places and transitions. Although Smith and Reisig (1987) use place and transition respecting morphisms, our morphisms are required to preserve arc-weights in addition. Second, Smith and Reisig (1987) discuss foldings of CE-systems into strict high-level nets. In this chapter we take a more concrete view of the process and give a construction to a specific class of high-level nets. Also the construction is given for non-strict nets, thus answering a question left open by Smith and Reisig (1987).

The first thing we need to do is to define the notion of folding. Since the intuition in the formation of a high-level net is that we are describing several identical processes with the same underlying net through one net it seems plausible to take the following as the definition.

Definition 4.1.1
A morphism $f : N_1 \to N_2$ in PetriG is a *folding* iff both f_T and f_P are surjective. □

Lemma 4.1.2
If $f : N_1 \to N_2$ is a folding then $N_{1/\ker f} \simeq N_2$. □

One of the advantages of working within the category PetriG is that the notion of surjection can readily be transfered from the category Set.

Proposition 4.1.3
Given an algebraic net AN the assignment

$$\begin{aligned} f_P(\langle p, a\rangle) &= \langle p, \perp\rangle \\ f_T(\langle t, \mathrm{ass}^{\#}_{A}\rangle) &= \langle t, \mathrm{ass}^{\#}_{\perp}\rangle \end{aligned}$$

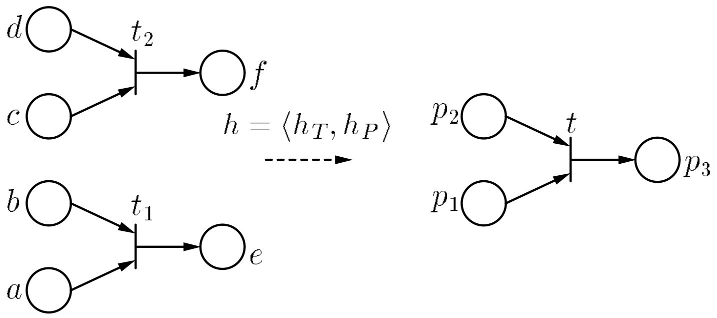
defines a folding $f : \mathsf{Unf}(AN) \to \mathsf{U}(AN)$.

Proof:
The fact that f_P and f_T are surjective follows directly from the definition. That f is a PetriG morphism is also clear. □

Proposition 4.1.4
Foldings define a natural transformation $\phi : \mathsf{Unf} \to \mathsf{U}$.

Proof:
Evident from the fact that the diagram

$$\begin{tikzcd} \mathsf{Unf}(AN) \arrow[r, "\phi(AN)"] \arrow[d, "\mathsf{Unf}(h)"'] & \mathsf{U}(AN) \arrow[d, "\mathsf{U}(h)"] \\ \mathsf{Unf}(AN') \arrow[r, "\phi(AN')"] & \mathsf{U}(AN') \end{tikzcd}$$



$$
\begin{array}{llcl}
h_T: & t_1 & \mapsto & t \\
 & t_2 & \mapsto & t \\
h_P: & a & \mapsto & p_1 \\
 & b & \mapsto & p_2 \\
 & c & \mapsto & p_1 \\
 & d & \mapsto & p_2 \\
 & e & \mapsto & p_3 \\
 & f & \mapsto & p_3
\end{array}
$$

Figure 4.1: A folding $h : N_1 \to N_2$ in PetriG.

commutes. $\square$

Thus each Algebraic net defines a unique folding. It is now natural to ask the converse question: Given a folding $f : N_1 \to N_2$ in PetriG does there exists an Algebraic net AN s.t. $\mathsf{Unf}(AN) \simeq N_1$ and $\mathsf{U}(AN) \simeq N_2$? The answer to this question is positive; the construction however is rather non-trivial.

Let us approach the question with an example. Consider the nets in figures 4.1 and 4.2. The map in figure 4.1 folds two independent transitions onto the same transition. The folded net can simulate the behavior of the original net, by having multiple markings on the places p_1 and p_2, but the information of which specific transition (t_1 or t_2) is fired is lost.

This information can be recovered by coding it into the annotation functions. Figure 4.2 gives one possible annotation for N_2. Its main virtue is that $\mathsf{Unf}(AN) \simeq N_1$. The idea is that for each equivalence class in $[p] \in P/\ker h$ and $[t] \in T/\ker h$ we construct an operation $f_{[t],[p]} : [t] \to [p]$ whose definition is such that the transition corresponding to $t_1 : a \otimes b \to e$ in $\mathsf{Unf}(AN)$ is fired with mode $\{x \leftarrow t_1\}$. The original behavior is thus recoverable. This suggests that there is a simple and straightforward way of coding the annotation directly from a given folding $f : N_1 \to N_2$. This is essentially the construction proposed by Smith and Reisig (1987).

However, the problem is more complicated. The construction as outlined above does not work if we fold several input-places of a transition into one. Consider the folding in figure 4.3. If we want to construct the annotation of the corresponding AN-net, we cannot simply take $f_{[t],[p_1]} : [t] \to [p_1]$ with $f_{[t],[p_1]}(t) = a \otimes b$, as then the underlying net would not be isomorphic to N_2, because it would have an arc-weight of 1 on the arc $p_1 \to t$. But if we construct two operations $f_{[t],a}$ and $f_{[t],b}$ with $f_{[t],a}(t) = a$ and $f_{[t],b}(t) = b$ then the input-weight function ι has the definition $\iota(t)(p_1) = f_{[t],a}(x) \otimes f_{[t],b}(x)$. This construction is formalized in the following theorem.

$$
\begin{array}{lll}
S: & \{p_1, p_2, p_3, t\} & \\
\Sigma: & & EQ: \\
& a :\to p_1 & f_{t,p_1}(t_1) = a \\
& b :\to p_2 & f_{t,p_1}(t_2) = c \\
& c :\to p_1 & f_{t,p_2}(t_1) = b \\
& d :\to p_2 & f_{t,p_2}(t_2) = d \\
& e :\to p_3 & f_{t,p_3}(t_1) = e \\
& f :\to p_3 & f_{t,p_3}(t_2) = f \\
& t_1 :\to t & \\
& t_2 :\to t & \\
& f_{t,p_1} : t \to p_1 & \\
& f_{t,p_2} : t \to p_2 & \\
& f_{t,p_3} : t \to p_3 &
\end{array}
$$

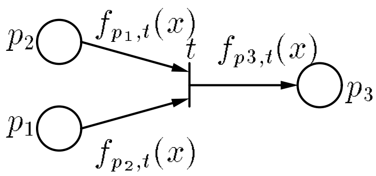
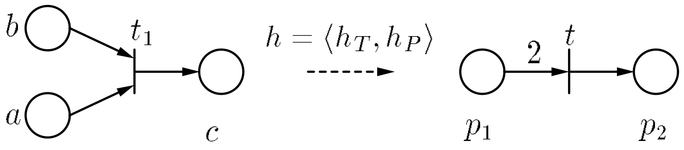


Figure 4.2: The AN-net corresponding to the folding in figure 4.1.



$$
\begin{array}{llll}
h_T: & t_1 & \mapsto & t \\
h_P: & a & \mapsto & p_1 \\
& b & \mapsto & p_1 \\
& c & \mapsto & p_2
\end{array}
$$

Figure 4.3: A folding $h' : N_1 \to N_2$ in <u>PetriG</u>.

Theorem 4.1.5
Given nets $N_1 = \langle P_1, T_1, \iota_1, o_1 \rangle$, $N_2 = \langle P_2, T_2, \iota_2, o_2 \rangle$ and a folding $f : N_1 \to N_2$ in <u>PetriG</u> there exists an Algebraic net AN s.t. $\mathsf{Unf}(AN) \simeq N_1$ and $\mathsf{U}(AN) \simeq N_2$.

Proof:
The proof is done in three stages. First we construct AN, then the isomorphism $\mathsf{Unf}(AN) \simeq N_1$, and finally the isomorphism $\mathsf{U}(AN) \simeq N_2$.

1. Let $AN = \langle ANS, A \rangle$ where $ANS = \langle S, \Sigma, EQ, X, T, P, \iota, o, \mathrm{sort} \rangle$ with S, P, T, X, Σ, EQ, ι, o and sort constructed as follows.

   (a) $P = P_2$, $T = T_2$, $S = P_2 \bigcup T_2$, where each sort $[t], [p]$ is inhabited by the elements of the equivalence relation, and $X = \{x_{[t]}\}_{[t] \in T_2}$.

   (b) sort is $\mathbf{id}_S \restriction P_2$.

   (c) To define the set of operator symbols, we first need to establish the domains and codomains of the operators. To this end we define functions $\delta_\iota : P_2 \times T_1 \to P_1$ and $\delta_o : P_2 \times T_1 \to P_1$ given by

   $$\delta_\iota([p], t) = \gamma(\iota(t)) \cap \gamma(f_P^{-1}([p]))$$

   and

   $$\delta_o([p], t) = \gamma(o(t)) \cap \gamma(f_P^{-1}([p])) \ .$$

   The functions give the set of input(output)-places of t in N_1 that are mapped to $[p]$ in N_2. For example in figure 4.1 $\delta_\iota(p_1, t_1) = \{a\}$, while in example 4.3 $\delta_\iota(p_1, t_1) = \{a, b\}$.
   Then let $n^i_{[t],[p]} = \max\{|\delta_i([p], t)| \mid t \in [t]\}$ for $i \in \{\iota, o\}$. $n^i_{[t],[p]} > 1$ if we fold several places of a transition onto one.
   The set of function symbols is given by[2]

   $$\begin{array}{rcl} \Sigma_f & = & \{f^i_{[t],[p],x} : [t] \to [p] \mid \\ & & [t] \in T, [p] \in P, 1 \leq x \leq n^i_{[t],[p]}\} \\ \Sigma_p & = & \{p :\to [p] \mid \forall p \in P_1\} \\ \Sigma_t & = & \{t :\to [t] \mid \forall t \in T_1\} \\ \Sigma & = & \Sigma_f \cup \Sigma_p \cup \Sigma_t \end{array}$$

   If $n^i_{[t],[p]} = 1$ we omit the x. So in figure 4.1

   $$\Sigma = \{f^\iota_{t,p_1}, f^\iota_{t,p_2}, f^o_{t,p_3}, a, b, c, d, t_1, t_2\},$$

[2] In the sequel i ranges over $\{\iota, o\}$

while in figure 4.3 we have

$$\Sigma = \{f^{\iota}_{t,p_1,1}, f^{\iota}_{t,p_1,2}, f^{o}_{t,p_2}, a, b, c, t_1\} \ .$$

(d) The construction of the equations is best explained by cases. If $n^{i}_{[t],[p]} = 0$ there is nothing to do. If $n^{i}_{[t],[p]} \geq 1$ the equations are defined by

$$EQ^i = \{f^{\iota}_{[t],x}(t) = y \mid \forall t, y : t \in [t] \wedge y \in \delta_i([p], t))\} \ .$$

In figure 4.3 we have that $f^{\iota}_{t,p_1,1}(t) = a \in EQ$.

(e) Then we have

$$\iota([t]) = \bigotimes_{1 \leq x \leq n^{\iota}_{[t],[p]}} \langle [p], f^{\iota}_{[t],[p],x}(x_{[t]})\rangle,$$

and

$$o([t] = \bigotimes_{1 \leq x \leq n^{o}_{[t],[p]}} \langle [p], f^{o}_{[t],[p],x}(x_{[t]})\rangle \ .$$

(f) Finally let $A = T_{\Sigma/\equiv}$.

2. The sets of places and transitions in $\mathsf{Unf}(AN)$ are given by

$$\begin{aligned} P' &= \bigcup_{p \in P} \{\{p\} \times [p]\} \ . \\ T' &= \{\langle [t], \mathrm{ass}^{\#}_{A}\rangle \mid [t] \in T, \mathrm{ass}^{\#}_{A} \in [Var([t]) \to A]\} \ . \end{aligned}$$

Because all function symbols are of the form $f : [t] \to [p]$ and by construction $Var([t]) = \{x_{[t]}\}$, the only valid assignments in this context are of the form $\{x_{[t]} \leftarrow t\}$ with $t \in [t]$. The set of transitions is thus isomorphic to T_{N_1}. Analogously for places.

The isomorphism $\mathsf{Unf}(AN) \simeq N_1$ is now obtained as follows. Because $P' \simeq P$ and $T' \simeq T$ we have the two morphisms $m_P : P' \to P : \langle p, q\rangle \mapsto q$ and $m_T : T' \to T : \langle [t], \{x_{[t]} \leftarrow t\}\rangle \mapsto t$ which form the morphism $\langle m_T, m_P\rangle : \mathsf{Unf}(AN) \to N_1$. It remains to show, that $\iota_{\mathsf{Unf}(AN)}; m^{\otimes}_{P} = m_T; \iota_{N_1}$. As a first step we have:

$$\begin{aligned} & m^{\otimes}_{P}(\iota_{\mathsf{Unf}(AN)}(\langle [t], \mathrm{ass}^{\#}_{T_\Sigma(X)}\rangle)) \\ &= m^{\otimes}_{P}(\mathrm{ass}^{*}_{T_{\Sigma/\equiv}}(\iota([t]))) \\ &= m^{\otimes}_{P}(\bigotimes_{1 \leq x \leq n^{\iota}_{[t],[p]}} \langle [p], \mathrm{ass}^{\#}_{T_{\Sigma/\equiv}}(f^{\iota}_{[t],[p],x})\rangle) \end{aligned}$$

Now, because each $\mathrm{ass}^{\#}_{T_{\Sigma/\equiv}}$ is of the form $\{x_{[t]} \leftarrow t\}$, and by definition each $f_{[t],[p],x}$ is a function, we have that $\mathrm{ass}^{\#}_{T_{\Sigma/\equiv}}(f_{[t],[p],x}) = y$ with $y \in \delta_\iota([p], t))$, where t is the t in $\{x_{[t]} \leftarrow t\}$. So we get:

$$\begin{aligned} m_P^{\otimes}(& \bigotimes_{1 \leq x \leq n^\iota_{[t],[p]}} \langle [p], \mathrm{ass}^{\#}_{T_{\Sigma/\equiv}}(f^\iota_{[t],[p],x}) \rangle) \\ = & \bigotimes_{1 \leq x \leq n^\iota_{[t],[p]}} \mathrm{ass}^{\#}_{T_{\Sigma/\equiv}}(f^\iota_{[t],[p],x}) \\ = & \bigotimes_{1 \leq x \leq n^\iota_{[t],[p]}} y_x \\ = & \iota_{N_1}(t) \\ = & \iota_{N_1}(t)(m_T(\langle [t], \mathrm{ass}^{\#}_{T_{\Sigma/\equiv}} \rangle)) \end{aligned}$$

The proof for o is analogous

3. To obtain the isomorphism $\mathsf{U}(AN) \simeq N_2$ observe that by construction we have for $\mathsf{U}(AN) = \langle T_U, P_U, \iota_U, o_U \rangle$ that $T_U = T_2$, and $P_U = P_2$.

   It remains to check that $\iota = \iota_2$ and $o = o_2$. By definition $\iota(t)_U = \iota; \mathrm{ass}^*_\perp = \bigotimes_{1 \leq x \leq n^\iota_{[t],[p]}} \langle [p] \rangle = \iota_2(t)$.

□

The construction codes the folding f into the annotation functions. Although we allow the sets T and P to be infinite, we have assumed that the functions f_T, f_P must be recursive. Thus according to a result by Bergstra and Tucker (1987) there exists a specification of the functions f_T, f_P.

To make the folding construction into a functor $\mathsf{Fold} : \underline{\mathsf{PetriG}} \to \underline{\mathsf{AN}}$, we need to define maps $\mathsf{Fold}_{\mathrm{obj}} : |\underline{\mathsf{PetriG}}| \to |\underline{\mathsf{AN}}|$ and $\mathsf{Fold}_{\mathrm{mor}} : \mathbf{Mor}_{\underline{\mathsf{PetriG}}} \to \mathbf{Mor}_{\underline{\mathsf{AN}}}$. However the construction is defined by a morphism, so it is not quite clear how to define the map on objects. Indeed what we have here is a comma-category construction, but we shall not delve into this question any deeper. An obvious solution would be to define $\mathsf{Fold}_{\mathrm{obj}}(N)$ as the folding of $f : N \to \perp$, where $\perp$ is the terminal object. But unfortunately $\underline{\mathsf{PetriG}}$ has no terminal object. The terminal object of $\underline{\mathsf{Petri}}$ is not usable, because the terminal maps are not maps in $\underline{\mathsf{PetriG}}$. The reason why $\underline{\mathsf{PetriG}}$ fails to have a terminal object is that since the arrows are free extensions of $\underline{\mathsf{Set}}$-morphisms, the total weight of the arc-annotation is always preserved.

However in $\underline{\mathsf{PetriG}}$ for each net N we can find an object N_M, such that the morphism $f : N \to N_M$ is unique. It has just one place, but for each possible combination of input and output weights a unique transition. Given a transition $t : a_1 \otimes \ldots \otimes a_n \to b_1 \otimes \ldots \otimes b_m$ let the type$(t) = \langle n, m \rangle$. Thus to each transition we can attach a pair of integers that denote its *type*.

Proposition 4.1.6
Given a net $N = \langle T, P, \iota, o\rangle$ there exists a net $N_M = \langle P_M, T_M, \iota_M, o_M\rangle$ with one place, $P_M = \{*\}$, and for each $t \in T$ there exist a unique $t' \in T_M$ such that type(t') = type(t), and the morphism $m_N : N \to N_M$ is unique.

Proof:
Clearly the function type partitions the set of transitions of N into equivalence classes. Thus f_T is uniquely defined. The map f_P is just the terminal map in $\underline{\mathsf{Set}}$. $\square$

Lemma 4.1.7
In $\underline{\mathsf{PetriG}}$, the nets N_M with one place form a partial order without bottom.

Proof:
Follows directly from the fact that the collection of nets N_M is isomorphic to $\mathcal{P}(\mathbb{N}_1 \times \mathbb{N}_1)$ ordered by inclusion, where $\mathbb{N}_1 = \{n \mid n \geq 1, n \in \mathbb{N}\}$. $\square$

Proposition 4.1.8
Given a $\underline{\mathsf{PetriG}}$-net N denote the Algebraic net obtained through proposition 4.1.6 by $\mathsf{Fold}(N)$. Given a $\underline{\mathsf{PetriG}}$-morphism $f : N \to N'$, there exists an $\underline{\mathsf{AN}}$-morphism $\mathsf{Fold}(f) : \mathsf{Fold}(N) \to \mathsf{Fold}(N')$ defining a functor $\mathsf{Fold} : \underline{\mathsf{PetriG}} \to \underline{\mathsf{AN}}$.

Proof:
The net-part of the map $\langle h_T, h_P, h_\Sigma, h_A\rangle : \mathsf{Fold}(N) \to \mathsf{Fold}(N')$ is obtained from the inclusion $i : N_M \to N'_M$.

The map h_Σ is constructed by "simulating" the map f by a signature morphism. This is possible, because in the process of folding, all the information of the net is retained in the equational signature.

On sorts it is given by $h_S = i_T \cup i_P$, and on variables by $h_X(x_{[t]}) = x_{h_T([t])}$ (it is injective because h_T is an injection). For the operators the situation is more complicated. Essentially the map is given by $h_\Sigma : f_{[t],[p],x} \mapsto f_{i_t([t]),i_p([p]),y}$, but is unclear what the x should map to. The problem arises because we must somehow recover the choice we made in step 1.(d) in theorem 4.1.5. However, we can recover the choice, because the set of equations of $\mathsf{Fold}(N)$ and $\mathsf{Fold}(N')$ have this mapping encoded. That is given the set of equations defining the behavior of an operator $f_{[t],[p],x}$ we can construct a map $\phi_{f_{[t],[p]}} : [p] \to \mathbb{N}$ given by $\phi_{f_{[t],[p]}}(p) = x$ iff $f_{[t],[p],x} = p \in EQ$. Now given $f_P : P \to P'$ we can

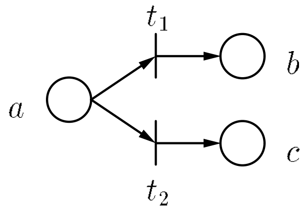
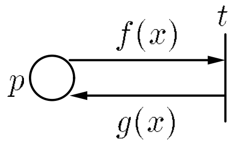


Figure 4.4: The net N.



$$
\begin{array}{lllll}
S: & \{p,t\} & & & \\
\Sigma: & a :\to p & & EQ & f(t_1) = a \\
& b :\to p & & & f(t_2) = a \\
& c :\to p & & & g(t_1) = b \\
& t_1 :\to t & & & g(t_2) = c \\
& t_2 :\to t & & & \\
& f : t \to p & & & \\
& g : t \to p & & &
\end{array}
$$

Figure 4.5: The folding of N.

construct a map $\Phi : \mathbb{N} \to \mathbb{N}$ such that the diagram commutes:

$$
\begin{tikzcd}
{[p]} \arrow[r, "f_P"] \arrow[d, "\phi_{f_{[t],[p]}}"'] & f_P([p]) \arrow[d, "\phi_{f_{f_T([t]),f_P([p])}}"] \\
\mathbb{N} \arrow[r, "\Phi"'] & \mathbb{N}
\end{tikzcd}
$$

The map h_Σ is then defined by:

$$h_\Sigma : f_{[t],[p],x} \mapsto f_{i_t([t]),i_p([p]),\Phi(x)} \, .$$

□

Corollary 4.1.9
$\mathsf{Unf}(\mathsf{Fold}(N)) \simeq N$

Proof:
Follows directly from theorem 4.1.5. □

Example 4.1.10
The net in figure 4.5 is the net corresponding to the folding $f : N \to N_T$ in figure 4.4. □

$$
\begin{array}{lll}
\texttt{sorts} & : & phil, fork \\
\texttt{opns} & : & ph_1, ph_2, ph_3 \quad :\rightarrow phil \\
 & & f_1, f_2, f_3 \quad\quad\;\; :\rightarrow fork \\
 & & l \quad\quad\quad\quad\quad\quad : phil \rightarrow fork \\
 & & r \quad\quad\quad\quad\quad\quad : phil \rightarrow fork \\
\texttt{eqns} & : & l(ph_i) = f_i \\
 & & r(ph_i) = f_i \quad\quad \text{for } i = 1, 2 \\
 & & r(ph_3) = f_1
\end{array}
$$

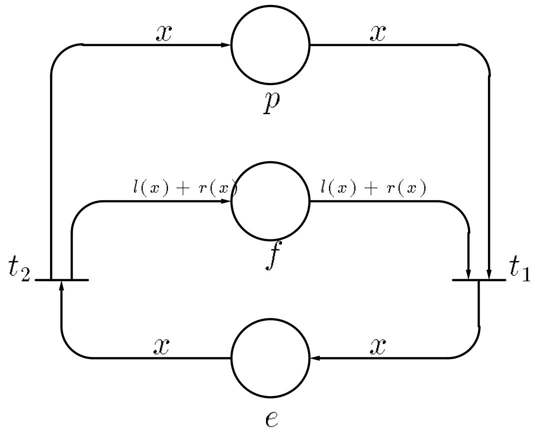


Figure 4.6: The specification AN of the dining philosophers problem.

Observe that choice is mapped into surjective annotation functions.

It is naturally interesting to ask whether $\mathsf{Fold}(\mathsf{Unf}(AN)) \simeq AN$? However it is rather easy to see, that this is not the case. We do not even have a map $\mathsf{Fold}(\mathsf{Unf}(AN)) \rightarrow AN$ as can be seen from figure 4.8 which is the result of folding the net in figure 4.7, which is the unfolding of the net in figure 4.6. It is clear that since all information of the specification in the net AN is lost during unfolding, there is no way to automatically recover this.

Here a folding has been defined as a surjective mapping between nets. Since a surjective map defines an equivalence on its domain, it would be interesting to look at the reverse question: When does an equivalence on places define a folding? Clearly the equivalence must be such that it respects the flow of the net.

Definition 4.1.11
Given a net $N = \langle T, P, \iota, o\rangle$ an equivalence relation $\equiv \in P \times P$ is said to be *compatible* iff

$$\forall\, [p], \forall\, t, t' \in i^{-1}([p]) : |\delta_i([p], t)| = |\delta_i([p], t')|,$$

where $i^{-1}([p]) = \{t \mid \exists\, p \in [p] : p \in \gamma(\delta_i(t))\}$ and $i \in \{\iota, o\}$. □

The intuition of this definition is that the equivalence relation should be

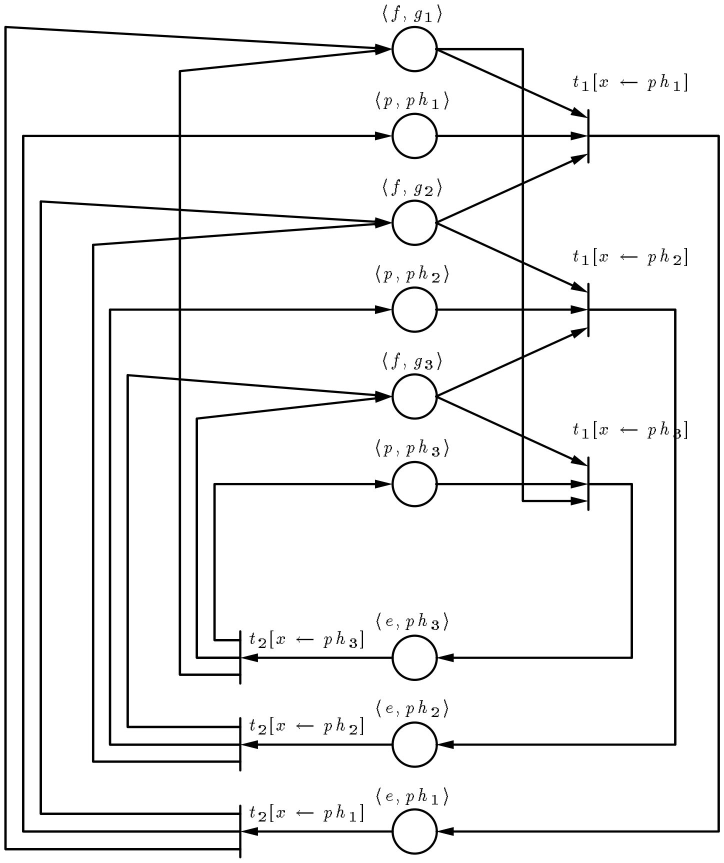


Figure 4.7: The net $\mathsf{Unf}(AN)$.

sorts : p,t_1,t_2

opns : $\langle f, g_1\rangle, \langle f, g_2\rangle, \ldots :\to p$
$\langle t_1, [x \leftarrow ph_1]\rangle, \ldots :\to t_1$
$\langle t_2, [x \leftarrow ph_1]\rangle, \ldots :\to t_2$
$f^{\iota}_{t_1,p,1} f^{\iota}_{t_1,p,2}, f^{\iota}_{t_1,p,3} : t_1 \to p$
$f^{o}_{t_1,p} : t_1 \to p$
$f^{\iota}_{t_2,p} : t_2 \to p$
$f^{o}_{t_2,p,1}, f^{o}_{t_2,p,2}, f^{o}_{t_2,p,3} : t_2 \to p$

eqns : $f^{\iota}_{t_1,p,1}(\langle t_1, [x \leftarrow ph_1]\rangle) = \langle f, g_1\rangle$
$f^{\iota}_{t_1,p,2}(\langle t_1, [x \leftarrow ph_1]\rangle) = \langle f, ph_1\rangle$
$f^{\iota}_{t_1,p,3}(\langle t_1, [x \leftarrow ph_1]\rangle) = \langle f, g_2\rangle$
$\vdots$

$$\iota(t_1) = f^{\iota}_{t_1,p,1}(x) \otimes f^{\iota}_{t_1,p,2}(x) \otimes f^{\iota}_{t_1,p,3}(x)$$
$$o(t_2) = f^{o}_{t_2,p,1}(x) \otimes f^{o}_{t_2,p,2}(x) \otimes f^{o}_{t_2,p,3}(x)$$

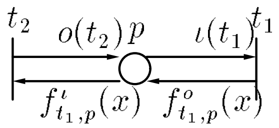


Figure 4.8: The result of $\mathsf{Fold}(\mathsf{Unf}(AN))$.

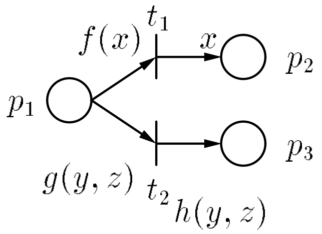


Figure 4.9: The net AN.

compatible with the arc-weights of the original net. We can now state the following proposition.

Proposition 4.1.12
Given a net $N = \langle T, P, \iota, o\rangle$ and a compatible equivalence relation $\equiv$, we can construct a folding $\langle f_T, f_P\rangle$, such that $\equiv = P/\ker f_P$. □

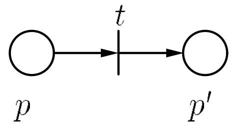
4.2 Foldings in AN

It is interesting also to look also at foldings in the category AN. That is given an Algebraic net AN a folding $f : \mathsf{U}(AN) \to N$, can we construct an Algebraic net AN' s.th $\mathsf{Unf}(AN) \simeq \mathsf{Unf}(AN')$ and $U(AN') \simeq N$?

This time it is not always possible to fold or to obtain a net with isomorphic semantics. Consider the Algebraic net AN in figure 4.9. Suppose we want to fold the transitions t_1 and t_2 into one. First of all we must make sure that the sort of places p_2 and p_3 are the same, otherwise it is not possible to construct an output-arc annotation for t. Let us suppose, that this is the case in this example. To construct the annotation functions of AN' we can use the idea of adding the set of transitions of AN as a new sort to the signature of AN'. One can then construct a new function k_t for each transition t of AN' that is defined by an equation $k_t^i(\overline{x}, t') = \pi_2(i(t'))$. So in our example we get the net in figure 4.10 (we have only written down the equations for the k_t^i, the rest of the specification is the same as for AN). However, now $\mathsf{Unf}(AN) \not\simeq \mathsf{Unf}(AN')$. To see why, notice that we have into k_t collected all the variables in $f(x)$ and $k(z,y)$. Thus we will for the case of t_1 have $Z \times Y$ assignments, where Z and Y are the sizes of the domains of the corresponding variables, which then implies that for each $t_1[x \leftarrow c]$ in the net $\mathsf{Unf}(AN)$ we have $Z \times Y$ transitions in $\mathsf{Unf}(AN')$. To get rid of this problem, we need to require that $Var(t) = Var(t')\ \ \forall t, t' \in [t]$.

Definition 4.2.1
Given an Algebraic net AN, a folding $f : \mathsf{U}(AN) \to N$ is an *AN-folding* iff:

1. $Var(t) = Var(t')\ \ \forall t, t' \in [t]$, and



$$\begin{array}{rcl} k_t^\iota(x,y,z,t_1) & = & f(x) \\ k_t^\iota(x,y,z,t_2) & = & g(y,z) \\ k_t^o(x,y,z,t_1) & = & x \\ k_t^o(x,y,z,t_2) & = & h(y,z) \end{array}$$

Figure 4.10: The net AN'.

2. $\text{sort}(p) = \text{sort}(p')\ \forall p, p' \in [p]$.

□

However there still remains a second problem. Suppose $\text{sort}(p_2) = \text{sort}(p_3)$ contains the elements $\{a, b\}$. Then $\mathsf{U}(AN)$ has the places $\{\langle p_2, a\rangle, \langle p_2, b\rangle, \langle p_3, a\rangle, \langle p_3, b\rangle\}$, while $\mathsf{U}(AN')$ only has $\{\langle [p], a\rangle, \langle [p], b\rangle\}$. The places in AN must thus be encoded in the annotations. This can be done by defining the sort of p' to be $[p'] \times \text{sort}_{AN}([p'])$. Then because $[p']$ will be inhabitated by $\{p_2, p_3\}$ we will obtain the correct number of places.

Then we can give the following theorem. Again, care must be taken in the case where two places are folded onto the same place.

Theorem 4.2.2
Given an Algebraic net AN and an AN-folding $f : \mathsf{U}(AN) \to N$, we can construct and Algebraic net AN' s.th. $\mathsf{Unf}(AN) \simeq \mathsf{Unf}(AN')$ and $U(AN') \simeq N$.
Proof:
Let $AN' = \langle S, \Sigma, EQ, X, T, P, \iota, o, \text{sort}\rangle$ be given by:

1. $P = P_N$,
2. $T = T_N$,
3. $X = X_{AN} \bigcup \{x_{[t]}\}_{[t] \in T_N}$,
4. $S = S_{AN} \bigcup T_N \bigcup \{\langle [p], s\rangle \mid [p] \in P_N \wedge s = \text{sort}_{AN}([p])\}$, and
5. $\text{sort}([p]) = \langle [p], \text{sort}_{AN}([p])\rangle$.
6. The set of operator symbols consists of the operator symbols of AN, together with a new constant operator for each transition in AN and the new operators that express the folding.

The set of transition symbols is

$$\Sigma_T = \{t :\to [t] \mid \forall t \in T_N\} \ .$$

To define the operators k we need the following auxiliary functions: $\delta_\iota : P_2 \times T_1 \to P_1$ and $\delta_o : P_2 \times T_1 \to P_1$ given by

$$\delta_\iota([p], t) = \gamma(\iota(t)) \cap \gamma(\mathsf{U}(f_P^{-1}([p])))$$

and

$$\delta_o([p], t) = \gamma(o(t)) \cap \gamma(\mathsf{U}(f_P^{-1}([p]))) \ .$$

The functions give the set of input(output)-places of t in AN that are mapped to $[p]$ in N.

Then let $n^i_{[t],[p]} = \max\{|\delta_i([p], t)| \mid t \in [t]\}$ for $i \in \{\iota, o\}$. $n^i_{[t],[p]} > 1$ if we fold several places of a transition onto one.

The set Σ_k is given by[3]

$$\begin{aligned} \Sigma_k \quad = \quad & \{k^i_{[t],[p],n} : \mathrm{sort}(Var(f^{-1}([t]))) \times [t] \to \mathrm{sort}([p]) \mid \\ & [t] \in T, [p] \in P, 1 \le n \le n^i_{[t],[p]}\}, \end{aligned}$$

So we have

$$\Sigma_{AN'} = \Sigma_{AN} \bigcup \Sigma_T \bigcup \Sigma_k \ .$$

7. The set of new equations is given by

$$EQ^i = \{k^i_{[t],[p],n}(\overline{x}, t) = \pi_n(i_{AN}(t)) \mid \forall t \in [t], [p] \in P, [t] \in T\}$$

where $\overline{x}$ is the set of all variables $Var(f^{-1}([t]))$ seen as a vector, and π_n picks the nth pair $\langle p, \hat{t}\rangle$ from the expression $\bigotimes_{i=1}^n \langle p_i, \hat{t}\rangle$. Thus

$$EQ_{AN'} = EQ_{AN} \bigcup EQ^i \ .$$

8. Then we have

$$\iota([t]) = \bigotimes_{1 \le x \le n^\iota_{[t],[p]}} \langle [p], f^\iota_{[t],[p],x}(x_{[t]})\rangle,$$

and

$$o([t] = \bigotimes_{1 \le x \le n^o_{[t],[p]}} \langle [p], f^o_{[t],[p],x}(x_{[t]}\rangle \ .$$

9. Finally $A' = T_{\Sigma_{AN'}/\equiv}$.

The fact that $N \simeq \mathsf{U}(AN')$ is evident. It remains to prove, that $\mathsf{Unf}(AN) \simeq$

[3] In the sequel i ranges over $\{\iota, o\}$

$\mathsf{Unf}(AN')$. We have:

$$\begin{aligned} P_{\mathsf{Unf}(AN')} &= \bigcup_{p \in P} \{\{p\} \times A'_{sort(p)}\}, \\ T_{\mathsf{Unf}(AN')} &= \{\langle [t], \mathrm{ass}^{\#}_{A_{AN'}} \rangle\} \end{aligned}$$

Now $P_{\mathsf{Unf}(AN')} \simeq P_{\mathsf{Unf}(AN)}$ by construction. The map $m_P : P_{\mathsf{Unf}(AN)} \rightarrow P_{\mathsf{Unf}(AN')}$ is given by $m_P(\langle [p], \langle p, \hat{t} \rangle \rangle) = \langle p, \hat{t} \rangle$ because of the definition of $k_{[t],[p],n}$ by the equations in EQ^i.

We also have that $T_{\mathsf{Unf}(AN')} \simeq T_{\mathsf{Unf}(AN)}$, because $Var(t) = Var(t') \quad \forall t, t' \in f^{-1}([t])$ and through the definition of EQ^i it is the case that

$$\mathrm{ass}^{\#}_{A_{AN'}} = \mathrm{ass}^{\#}_{A_{AN}} \bigcup \{x_{[t]} \leftarrow t\},$$

where $\mathrm{ass}^{\#}_{A_{AN}} \in [Var(t) \rightarrow A_{AN}]$. The map $m_T : T_{\mathsf{Unf}(AN)} \simeq T_{\mathsf{Unf}(AN')}$ is now given by

$$m_T(\langle t, \mathrm{ass}^{\#}_{A_{AN}} \rangle) = \langle [t], \mathrm{ass}^{\#}_{A_{AN}} \bigcup \{x_{[t]} \leftarrow t\} \rangle \ .$$

It remains to show that $\iota_{\mathsf{Unf}(AN)}; h_P^{\otimes} = h_T; \iota_{\mathsf{Unf}(AN')}$, where $h_P = m_P^{-1}$ and $h_T = m_T^{-1}$:

$$\begin{aligned} & h_P^{\otimes}(\iota_{\mathsf{Unf}(AN)}(\langle [t], \mathrm{ass}^{\#}_{T_{\Sigma_{AN'}}} \rangle)) \\ &= h_P^{\otimes}(\mathrm{ass}^{*}_{T_{\Sigma_{AN'}}}(\iota([t]))) \\ &= h_P^{\otimes}(\bigotimes_{1 \le x \le n^{\iota}_{[t],[p]}} \langle [p], \mathrm{ass}^{\#}_{T_{\Sigma_{AN'}}}(k^{\iota}_{[t],[p],x}) \rangle) \end{aligned}$$

Now, because each $\mathrm{ass}^{\#}_{T_{\Sigma_{AN'}}(X)}$ is of the form $\mathrm{ass}^{\#}_{A_{AN}} \bigcup \{x_{[t]} \leftarrow t\}$, and by definition each $k_{[t],[p],x}$ is a function, we have that $\mathrm{ass}^{\#}_{T_{\Sigma_{AN}}}(k_{[t],[p],x}) = \pi_n(\iota_{AN}(t))$, where t is the t in $\{x_{[t]} \leftarrow t\}$. So we get:

$$\begin{aligned} & h_P^{\otimes}(\bigotimes_{1 \le x \le n^{\iota}_{[t],[p]}} \langle [p], \mathrm{ass}^{\#}_{T_{\Sigma_{AN}}}(k^{\iota}_{[t],[p],x}) \rangle) \\ &= h_P^{\otimes}(\bigotimes_{1 \le x \le n^{\iota}_{[t],[p]}} \langle [p], \pi_n(\iota_{AN}(t)) \rangle, \\ &= \bigotimes_{1 \le x \le n^{\iota}_{[t],[p]}} \pi_n(\iota_{AN}(t)) \\ &= \iota_{AN}(t) \\ &= \iota_{\mathsf{Unf}(AN)}(h_T(\langle t, \mathrm{ass}^{\#}_{T_{\Sigma_{AN}}} \rangle)) \end{aligned}$$

$\square$

It turns out, that since every Petri net is an Algebraic net through the functor G, the construction of theorem 2.1.12 is a special case of the construction presented above.

4.3 Deadlock preserving skeletons

The idea of forgetting the annotations on the Algebraic net and investigating the resulting *skeleton* for deadlocks was first proposed by Vautherin (1987). As pointed out by Findlow (1991), the result is of limited usefulness, as even if the dead skeletal marking is reachable its pre-image may not be. Vautherin (1987) gave a sufficient condition on the Algebraic net to have the property that a marking is dead iff the image of the marking in the skeleton is dead, but this condition is very restrictive. This observation later led to a set of necessary and sufficient conditions for a net to have a deadlock preserving skeleton (Findlow 1992), together with a transformation algorithm that unfolds and then partially refolds the net until it satisfies the conditions. In this section we will transfer Findlows' result to Algebraic nets to give an application for the construction in section 4.1.

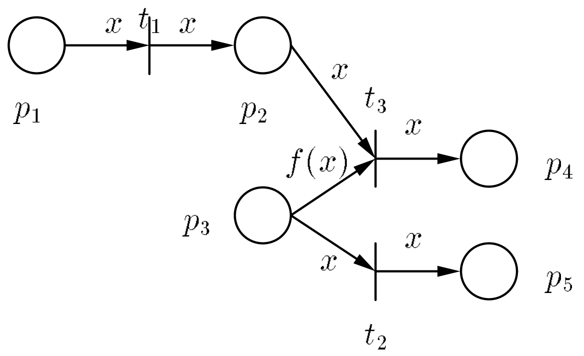
The problem with the original idea of analyzing skeletons, as pointed out by Findlow (1991), is that since it is defined without reference to an initial marking it does not distinguish between reachable dead markings and non-reachable dead markings. Thus if we look at the skeleton $\mathsf{U_N}(AN)$ and find a deadlock, it may well be, that a corresponding marking in the net AN is not reachable. To overcome this problem Vautherin (1987) proposed the following condition.

Proposition 4.3.1
Given an Algebraic net $\langle ANS, T_{\Sigma/\equiv}\rangle$ if for each transition t and each place p in ${}^\bullet t$, with $\iota(t) = \langle p, f_1\rangle \otimes \ldots \otimes \langle p, f_n\rangle$ we have:

1. $(p, i) \neq (p', j) \Rightarrow Var(f_i) \bigcap Var(f_j) = \varnothing$, and
2. $\alpha(f_i)$ is surjective in $T_{\Sigma/\equiv}$,

then each marking m is a deadlock in $\mathsf{Unf}(\langle ANS, T_{\Sigma/\equiv}\rangle)$ iff m is a deadlock in $\mathsf{Unf}(\langle ANS, \bot\rangle)$ (the skeleton). $\square$

Condition 1 states that each arc must have a distinct set of variables, and condition 2 that for every input place, the corresponding arc annotation has the whole sort of this places as its domain. Together these conditions are sufficient to ensure that a marking is enabled irrespective of the identity of the tokens. However these conditions are very restrictive and there are many examples where the net has a deadlock-preserving skeleton although it does not satisfy the conditions. For example the net N in figure 4.11 has a skeleton whose every deadlock is also a deadlock in N although it does not satisfy Vautherins conditions. The fact that transition t_3 does not satisfy the condition is circumvented by transition t_2 which fires in any mode.



$$\begin{array}{rll} sort: & s & \\ ops: & a: & \rightarrow s \\ & b: & \rightarrow s \\ & f: & s \rightarrow s \\ eqns: & f(a) & = a \\ & f(b) & = a \end{array}$$

Figure 4.11: A net with DPS that does not satisfy Vautherins condition

The basic idea in Findlows method is to test whether a place is live at a marking regardless of its color. In our case this amounts to the following definitions.

Definition 4.3.2
For any marking M of $\mathsf{Unf}(AN)$, with a token in place p, let $M(\langle p,\hat{t}\rangle \leftarrow \langle p,\hat{t}'\rangle)$ be the marking obtained by replacing the token $\hat{t}$ in p with $\hat{t}'$. Now define $\langle p,\hat{t}\rangle \leftarrow \langle p,\hat{t}'\rangle$ iff for any marking M of $\mathsf{Unf}(AN)$ M is live and $\hat{t} \in M(p) \Rightarrow M(\langle p,\hat{t}\rangle \leftarrow \langle p,\hat{t}'\rangle)$ is live. We will say that $\langle p,\hat{t}\rangle$ is interchangeable with $\langle p,\hat{t}'\rangle$ and write $\langle p,\hat{t}\rangle \leftrightarrow \langle p,\hat{t}'\rangle$ iff $\langle p,\hat{t}\rangle \leftarrow \langle p,\hat{t}'\rangle$ and $\langle p,\hat{t}'\rangle \leftarrow \langle p,\hat{t}\rangle$. □

Proposition 4.3.3
The relation $\langle p,\hat{t}\rangle \leftrightarrow \langle p,\hat{t}'\rangle$ defines an equivalence relation on the places of $\mathsf{Unf}(AN)$, denoted by $P_{\mathsf{Unf}(AN)}/\leftrightarrow$. □

The following lemma is now easy to see, because only places that are instances of the same high-level place are in the relation $\leftrightarrow$.

Lemma 4.3.4
$\leftrightarrow$ is a compatible equivalence relation. $\square$

The necessary and sufficient condition is then given by the following proposition.

Proposition 4.3.5
Let N be and Algebraic net. The following is a necessary and sufficient condition for every dead marking of N to have a dead skeletal image:

- For any place p of N, and any terms $\hat{t}, \hat{t}'$ of $sort(p)$

$$\langle p, \hat{t} \rangle \leftrightarrow \langle p, \hat{t}' \rangle$$

$\square$

Findlow (1992) proposes an algorithm for calculating the equivalence relation $\leftrightarrow$.

We can now obtain a version of the net AN that has a deadlock-preserving skeleton through the following.

Theorem 4.3.6
Let $AN = \langle S, \Sigma, EQ, X, T, P, \iota, o, \mathrm{sort}, T_\Sigma \rangle$ be an arbitrary Algebraic net. Define the net AN' as the net obtained by folding along

$$\langle f_P, f_T \rangle : \mathsf{Unf}(AN) \rightarrow \mathsf{Unf}(AN)/\leftrightarrow.$$

Then AN' has a deadlock-preserving skeleton.

Proof:
Because of lemma 4.3.4 we know that AN' is well defined and its unfolding is isomorphic to $\mathsf{Unf}(AN)$. Then the result follows because of proposition 4.3.5. $\square$

It should also be noted that since we are only interested in the skeleton, it is actually unnecessary to construct the specification for AN.

In this chapter we have presented a construction that given a folding returns an Algebraic net whose underlying net is isomorphic to the codomain of the folding and whose unfolding is isomorphic to the domain of the folding. We have also shown that this construction gives rise to a functor from the category <u>PetriG</u> to the category AN, and presented and application for the construction.

In this chapter we solved the problem of describing foldings of non-strict nets left open by Smith and Reisig (1987), although in a very concrete manner.

Due to the non-strictness of our nets and the algebraic nature of the morphism in PetriG we are unable to come up with a classification scheme of morphisms and the corresponding classification of high-level systems as proposed by Smith and Reisig (1987). Our construction gives just one type of annotation functions (unary-functions). This is also the reason why the folding and unfolding functors are not adjoint. There is no way to relate the specification obtained by folding to the original specification since all information about it is lost while unfolding.

A construction similar to the one in section 4.2 has recently also been proposed by Battiston, Cindio, Mauri and Rapanotti (1991) in the context of so called minimal models of OBJSA nets. They present a construction that maps an OBJSA net into an equivalent net that is minimal in the sense that it has only one place for each net-component. This mapping allows them to define an equivalence on OBJSA nets that describes the level of abstraction of the net. Our idea of partitioning PetriG according to the types of the transitions was inspired by the minimal model construction. However our folding construction is different because we fold transitions. The minimal model construction leaves the transitions alone and only folds the places of each transition.

It should be noted that there are other ways to define the notion of AN-folding in section 4.2. The requirement that the sorts should be preserved could be circumvented by introducing new "union" sorts for each place in the folded net. Then the equations for the operators $k^i_{[t],[p],x}$ would have to be modified to include suitable "injection" operators.

Our construction also has relied heavily on the fact that the morphisms in PetriG are free extensions of functions. It would be interesting to look at the construction when the morphisms are allowed to be Mon-morphisms. However we restricted ourselves to PetriG, since it is not quite clear what the right notion of folding is in Petri.

Chapter 5

Linear logic

Linear logic was discovered by J-Y. Girard (1987a) while trying to extend his coherent semantics, a domain theoretic model of second-order lambda-calculus (Girard 1986), to a sum of types construct. The sum of types is the mathematical analogue to the union types of programming languages and its treatment in denotational semantics is troublesome. Indeed this was also the case with coherent semantics, but as a side product linear logic was discovered. The name linear stems, according to Girard (1989b), from the linear algebra like nature of some of the connectives of linear logic.

This special nature of the connectives gives linear logic a resource conscious character and linear logic has gained considerable interest from the computer science community. Applications include:

- implementation models for functional languages (Girard and Lafont 1987, Lafont 1988, Wadler 1990),
- the study of process algebras (Abramsky and Vickers 1990, Abramsky 1988, Abadi and Plotkin 1991),
- logic programming (Andreoli and Pareschi 1990, Cerrito 1990, Harland and Pym 1990),
- planning in AI (Masseron, Tollu and Vauzeilles 1990),
- and net theory (Asperti, Gorrieri and Ferrari 1990, Brown 1989a, Brown 1989b, Brown and Gurr 1990, Brown 1991, Engberg and Winskel 1994, Gunter and Gehlot 1989, Marti-Oliet and Meseguer 1989).

The list is by no means meant to be comprehensive.

The applications relevant to our purposes are the applications to net theory. The main approaches can be classified as:

- proof theoretic (Brown 1989b, Gunter and Gehlot 1989),
- model theoretic (Brown 1989a, Engberg and Winskel 1994),
- and category theoretic (Brown and Gurr 1990, Marti-Oliet and Meseguer 1989).

In the proof theoretic approach the idea is to try to relate computations in nets to proofs of linear logic formulae. Brown (1989b) equates a net to a formula of linear logic and proves that reachability corresponds to provability in linear logic. On the other hand Gunter and Gehlot (1989) view a net as a theory. They are able to relate the cut-elimination of a proof (c.f. (Gallier 1987) sec. 6) to the notion of maximally concurrent process.

The model theoretic approaches of (Brown 1989a) and Engberg and Winskel (1994) are essentially the same. The idea is to generate a model of linear logic from the behaviors of a net.

The category theoretic approaches can really not be compared as their aims are completely different. The idea in (Marti-Oliet and Meseguer 1989) is to take the categories of nets of Meseguer and Montanari and close them under the new operators of linear logic. The approach of Brown and Gurr (1990) is based on the Dialectica Categories of Valeria de Paiva (de Paiva 1989a, de Paiva 1989b) which are a category theoretic model of Gödels "Dialectica interpretation" of higher order arithmetic (Gödel 1958). The constructs available in these categories give rise to very interesting constructions on nets that can be interpreted as linear logic connectives. The approach could be summed up in the slogan "nets are linear logic propositions". Unfortunately the approach currently only works with elementary nets.

In this chapter we will extend the model theoretic approach of Brown (1989a) to cover Algebraic Nets. We first show that the construction presented by Brown extends to a functor, and by taking the composition of this functor with the Unf-functor we obtain a model of linear logic from an Algebraic net. The results presented in this chapter have been reported in (Lilius 1991) and (Lilius 1992).

5.1 Syntax and proof-theory

Modern introductions to linear logic like appendix B in (Girard, Lafont and Taylor 1989) introduce linear logic through the sequent calculus; as this is to date the most accessible way to understanding linear logic. Essentially the different flavors of linear logic (intuitionistic, classical and predicate) are obtained by deleting the contraction and weakening rules from the standard

Axiom:
$$\frac{}{A \vdash A}\,(\mathrm{Id})$$

Structural Rule:
$$\frac{\Lambda, A, B, \Delta \vdash C}{\Lambda, B, A, \Delta \vdash C}\,(\mathrm{Exchange})$$

Cut Rule:
$$\frac{\Lambda \vdash A \quad A, \Delta \vdash B}{\Lambda, \Delta \vdash B}$$

Logical Rules:

$$\frac{}{\vdash 1}\,(1\text{ - R}) \qquad \frac{\Lambda \vdash A}{\Lambda, 1 \vdash A}\,(1\text{ - L})$$

$$\frac{\Lambda \vdash A \quad \Delta \vdash B}{\Lambda, \Delta \vdash A \otimes B}\,(\otimes\text{ - R}) \qquad \frac{\Lambda, A, B \vdash C}{\Lambda, A \otimes B \vdash C}\,(\otimes\text{ - L})$$

$$\frac{\Lambda, A \vdash B}{\Lambda \vdash A \multimap B}\,(\multimap\text{ - R}) \qquad \frac{\Lambda \vdash A \quad B, \Delta \vdash C}{\Lambda, A \multimap B, \Delta \vdash C}\,(\multimap\text{ - L})$$

$$\frac{\Lambda \vdash A \quad \Lambda \vdash B}{\Lambda \vdash A \& B}\,(\&\text{ - R}) \qquad \frac{\Lambda, A \vdash C}{\Lambda, A\&B \vdash C} \quad \frac{\Lambda, B \vdash C}{\Lambda, A\&B \vdash C}\,(\&\text{ - L})$$

$$\frac{\Lambda \vdash A}{\Lambda \vdash A \oplus B} \quad \frac{\Lambda \vdash B}{\Lambda \vdash A \oplus B}\,(\oplus\text{ - R}) \qquad \frac{\Lambda, A \vdash C \quad \Lambda, B \vdash C}{\Lambda, A \oplus B \vdash C}\,(\oplus\text{ - L})$$

$$\frac{!\Lambda \vdash A}{!\Lambda \vdash !A}\,(!\text{ - R}) \qquad \frac{\Lambda, A \vdash B}{\Lambda, !A \vdash B}\,(\mathrm{Dereliction})$$

$$\frac{\Lambda, !A, !A \vdash B}{\Lambda, !A \vdash B}\,(\mathrm{Contraction}) \qquad \frac{\Lambda \vdash B}{\Lambda, !A \vdash B}\,(\mathrm{Weakening})$$

$$\frac{\Lambda, A[a/x] \vdash B}{\Lambda, \bigvee x\,.\,A \vdash B}\,(\textstyle\bigvee\text{ - L}) \qquad \frac{\Lambda \vdash A}{\Lambda \vdash \bigvee x\,.\,A}\,(\textstyle\bigvee\text{ - R})$$

$$\frac{\Lambda, A \vdash B}{\Lambda, \bigwedge x\,.\,A \vdash B}\,(\textstyle\bigwedge\text{ - L}) \qquad \frac{\Lambda \vdash A[a/x]}{\Lambda \vdash \bigwedge x\,.\,A}\,(\textstyle\bigwedge\text{ - R})$$

Figure 5.1: Sequent calculus for intuitionistic predicate LL.

sequent calculus formulations of the corresponding logics. In figure 5.1 the sequent calculus formalization of linear intuitionistic predicate logic is given.

When comparing the formulation in figure 5.1 to formulations of intuitionistic predicate logic one can immediately see that the effect of deleting the two structural rules of contraction and weakening is significant. None of the standard rules for the logical connectives ($\wedge, \vee, \Rightarrow$) are present; they have all been replaced by new connectives. We will try to give intuitive interpretations in terms of the behavior of a net to all these connectives in subsection 5.4. Below we shall just give an informal description of the intuitionistic version of the calculus. The new connectives are classified as *multiplicatives* ("tensor" $\otimes$ and "linear implication" $\multimap$), *additives* ("tensor sum" $\oplus$ and "direct sum" $\&$) and *exponentials* ("of course" $!$). The distinction between multiplicatives and additives is best seen in the sequent calculus by noticing that the additives always use the *same* context, while the multiplicatives are used to combine different contexts. On an intuitive level the need for the large number of connectives can be understood as follows. Take the introduction rule for the and $\wedge$ connective in the sequent calculus of intuitionistic logic as given in (Girard et al. 1989):

$$\frac{\Lambda, A \vdash C}{\Lambda, A \wedge B \vdash C}\,(\mathcal{L}1\wedge)$$

The correctness of this rule is based on the interpretation of a sequent $\Lambda \vdash C$ as *the conjunction of all the premises in* Λ *entail* C. Thus in the presence of weakening the above rule is actually an abbreviation for the following proof:

$$\frac{\dfrac{\Lambda, A \vdash C}{\Lambda, A, B \vdash C}\,(\text{Weakening})}{\Lambda, A \wedge B \vdash C}\,.$$

Indeed there exist sequent calculus formulations of classical and intuitionistic logic that take this approach and simply have one introduction rule for $\wedge$ on the left (c.f. (Gallier 1987)). Now it is easy to see that without weakening we have to postulate a rule that allows the introduction of conjuncts on the left. But, as we have deleted weakening, we want to give the connective a new name, so that this implicit use of weakening can be distinguished from other uses. Thus the analogue to our rule above is the rule (&-L) in figure 5.1. On the other hand the rule

$$\frac{\Lambda \vdash A \quad \Delta \vdash B}{\Lambda, \Delta \vdash A \wedge B}\,(\mathcal{R}\wedge)$$

that introduces conjunction of the right does not implicitly use the weakening rule. This is "represented" in figure 5.1 by the rule ($\otimes$-R). Another way of saying this is that contraction and weakening allow one to think of the list of premises as sets, while in linear logic premises are to be treated as multisets.

The removal of the contraction and weakening rules makes the fragment with the multiplicatives and additives strictly weaker than intuitionistic logic. To get the strength of intuitionistic logic back the exponential connective "of course" is introduced. The idea is that we can allow weakening and contraction, but that we, as previously, want to mark the use, explicit or implicit, of weakening. The rules of weakening and contraction are put back in the logic as logical rules for the connective "of course". The sequent calculus in figure 5.1 lacks rules for negation. Linear negation, that is to be distinguished from intuitionistic negation, is introduced by first fixing a logical constant $\perp$ denoting linear absurdity and then defining $A^{\perp} = A \multimap \perp$. Using linear negation intuitionistic propositional logic can now be encoded into intuitionistic linear logic. One example of such a coding is the translation

$$A \wedge B = A \& B \quad A \vee B = {!}A \oplus {!}B \quad A \Rightarrow B = {!}A \multimap B \quad \neg A = {!}A \multimap \perp .$$

Other translations are given in (Girard 1987a). They all have different properties in the sense that they enlighten different aspects about provability in intuitionistic logic.

The remaining connectives are the quantifiers. Their names are *any* $\bigvee$ and *some* $\bigwedge$. Girard gives their definition in semantic terms as infinite generalizations of the additives. The quantifiers have been studied by Girard (1987b), but only their proof-theoretic properties are investigated. We shall see in section 5.4 how they can be used in modeling individual tokens in high-level nets.

To talk about properties of the net we introduce typed intuitionistic predicate linear logic. The development of the language follows (Goguen and Burstall 1990).

Definition 5.1.1
A *first order signature* Ω is a triple $\Omega = \langle S, \Sigma, \Pi \rangle$ where:

1. S is a set of *sorts*,
2. Σ is an $S^* \times S$-indexed family of sets of *operator* or *function* symbols, and
3. Π is an S^*-indexed family of sets of *predicate* or *relation* symbols.

□

The definition of sentences over a first-order signature Ω requires the following auxiliary definitions. Let $\mathcal{X}$ be a fixed infinite set of variable symbols, and let $X : \mathcal{X} \to S$ be a partial function i.e., sort assignment. We will also think of X as the union of the sets $X_s = \{x \in \mathcal{X} | X(x) = s\}$. Define the S-indexed family $TERM_X(\Omega)$ of (Ω, X)-*terms* to be the carriers of $T_\Sigma(X)$, the free Σ-algebra with generators X. Define the S-indexed function $Free$ on $TERM_X(\Omega)$ inductively by:

1. $Free_s(x) = x$ for $x \in X_s$, and
2. $Free_s(\sigma(t_1, \ldots, t_n)) = \bigcup_{i=1}^{n} Free(t_i)$.

Finally define $TERM(\Omega)$ to be the disjoint union of all $TERM_X(\Omega)$. This means that we always know the variables and their type in a term.

Definition 5.1.2
A *well-formed* (Ω, X)*-formula* is an element of the carrier of the (one-sorted) free algebra $WFF_X(\Omega)$ having *atomic* (Ω, X)*-formulae*

$$\{\pi(t_1, \ldots, t_n) | \pi \in \Pi_u \quad \text{with} \quad u = s_1 \ldots s_n \quad \text{and} \quad t_i \in TERM_X(\Omega)_{s_i}\}$$

as generators, and having the following one-sorted signature:

1. constants $1, 0, \mathbf{I}$,
2. binary infix operators $\otimes, \oplus, \&, \multimap$, and
3. unary prefix operators $(\bigvee x)$ and $(\bigwedge x)$ for each $x \in X$.

Let $WFF(\Omega)$ be the union of all $WFF_X(\Omega)$.

The functions Var and $Free$ that give the set of *variables* and *free variables* that are used in Ω-formulae, are defined inductively by

1. $Var(c) = Free(c) = \varnothing$ for $c \in \{1, 0, \mathbf{I}\}$,
2. $Var(\pi(t_1, \ldots, t_n)) = Free(\pi(t_1, \ldots, t_n)) = \bigcup_{i=1}^{n} Free_{s_i}(t_i)$,
3. $Var(A \Box B) = Var(A) \cup Var(B)$, and $Free(A \Box B) = Free(A) \cup Free(B)$ with $\Box \in \{\otimes, \oplus, \&, \multimap\}$,
4. $Var((\bigvee x)P) = Var((\bigwedge x)P) = Var(P) \cup \{x\}$,
and $Free((\bigvee x)P) = Free((\bigwedge x)P) = Free(P) - \{x\}$.

An Ω*-sentence* is a *closed* Ω-formula, that is, an Ω-formula P for which $Free(P) = \varnothing$.

The *typed linear intuitionistic predicate language* over a first order signature Ω is the set of all Ω-sentences and it is denoted L_Ω. When the context is clear the Ω shall be omitted. □

The sequent calculus formulation of the axiom system of linear intuitionistic predicate calculus was given in figure 5.1. To accommodate the typed version of linear intuitionistic predicate calculus, the rules ($\bigvee$ - L), ($\bigvee$ - R), ($\bigwedge$ - L), and ($\bigwedge$ - R) need to be modified by adding a type to the variable x.

$$\frac{\Lambda, A[a^s/x^s] \vdash B}{\Lambda, \bigvee x^s \,.\, A \vdash B}\,(\bigvee \text{ - L}) \qquad \frac{\Lambda \vdash A}{\Lambda \vdash \bigvee x^s \,.\, A}\,(\bigvee \text{ - R})$$

$$\frac{\Lambda, A \vdash B}{\Lambda, \bigwedge x^s \,.\, A \vdash B}\,(\bigwedge \text{ - L}) \qquad \frac{\Lambda \vdash A[a^s/x^s]}{\Lambda \vdash \bigwedge x^s \,.\, A}\,(\bigwedge \text{ - R})$$

5.2 Quantale semantics

Quantales are models of linear logic. They can be seen as an algebraic axiomatization of the sequent calculus rules of linear logic, in the same vein as boolean algebras are an algebraic axiomatization of classical logic.

Quantales were originally introduced by Mulvey (1986) in an attempt to cast light on the connections between C^*-algebras and quantum mechanics. The work by Mulvey, and Girard's attempts to give a semantics of linear logic in terms of C^*-algebras (Girard 1989a, Girard 1990), inspired several authors to study quantales and their relation to linear logic (Abramsky 1988, Yetter 1990).

A quantale is a complete lattice enriched with a monoidal operation.

Definition 5.2.1
A commutative quantale Q is a quadruple $\langle Q, \leq, \otimes, \mathbf{I} \rangle$ such that:

- $\langle Q, \leq \rangle$ is a complete join semi-lattice with top ($\top$) and bottom ($\bot$),
- $\langle Q, \otimes, \mathbf{I} \rangle$ is a commutative monoid, and
- the monoidal operation ('tensor') distributes over joins, i.e. for J an indexing set:
$$a \otimes \bigvee_{j \in J} b_j = \bigvee_{j \in J} (a \otimes b_j) \ .$$

□

Every monoid gives rise to a *free* quantale as shown by the following example.

Example 5.2.2
Let $N = \langle T, P, \iota, o \rangle$ be a Petri net. Take the set of all markings of then net $mark(N)$. The markings of the net are elements of a monoid (the monoidal operation will be denoted by $+$ to distinguish it from the operation in the quantale). We can easily extend this monoidal operation to sets of markings, i.e. for $P, Q \subseteq mark(N)$ define

$$P + Q = \{p + q | p \in P, q \in Q\} \ .$$

Let $\mathcal{P}(mark(N))$ be the powerset of the markings of N and $\underline{0}$ be the zero marking i.e. the unit of the monoid P^+. Then

$$\langle \mathcal{P}(mark(N)), +, \{\underline{0}\} \rangle$$

is a commutative monoid. Finally because of the natural subset ordering on the set $\mathcal{P}(mark(N))$ we have that the tuple

$$q(N) = \langle \mathcal{P}(mark(N)), \subseteq, +, \{\underline{0}\} \rangle$$

is a commutative quantale. Furthermore the assignment

$$q : N \mapsto \langle \mathcal{P}(mark(N)), \subseteq, +, \{\underline{0}\} \rangle$$

defines a function from $|\underline{\mathsf{PetriG}}|$ to $|\underline{\mathsf{Quant}}|$. □

As we shall be working with maps between quantales we need a notion of morphism for quantales.

Definition 5.2.3
A morphism of quantales is a function $f : Q_1 \to Q_2$ that is monotonic and preserves $\bigvee$, $\otimes$, and $\mathbf{I}$. □

Using this definition of a morphism the following is obvious.

Proposition 5.2.4
Quantales and their morphism form a category $\underline{\mathsf{Quant}}$. □

5.3 Quantales and nets

In this section we shall show how given a net we can construct a quantale that describes the behavior of the net.

The construction described in this section was independently discovered by-Brown (1991) and Engberg and Winskel (1994). The intuitive idea underlying the construction is that linear implication should correspond to reachability of markings. General formulas of linear logic will then be statements about the reachability of markings.

To construct a model of linear logic from a net we shall proceed as follows. First from the set of markings of the net $mark(N)$ the *free* quantale $q(N)$ over $mark(N)$ is generated. Then we prove that the reachability relation between markings defines a special kind of map between quantales, a quantic nucleus, on the quantale $q(N)$. Finally we will use a result by Niefield and Rosenthal (1988) to obtain our model. In the model the interpretation of an atomic proposition p is the set of all markings that are reachable from the corresponding place p in the net (Brown 1991) or, the set of all markings that are a prerequisite for the fact that place p become marked (Engberg and Winskel 1994). Our contribution is the proof that both constructions yield a functor from the category $\underline{\mathsf{PetriG}}$ to the category of quantales $\underline{\mathsf{Quant}}$. The omitted proofs can be found in chapter 7 of (Brown 1991), and (Niefield and Rosenthal 1988).

Recall from example 5.2.2 that net N each defines a quantale $q(N)$. This quantale is however not the quantale we want, because its elements are arbitrary sets of markings and thus these sets do not represent the behavior of the net. What we need to do, is to narrow down the allowed sets of markings so that in a set M, with $a, b \in M$, either a is reachable from b or b is reachable from a. Then the subset ordering of these sets of markings represents the reachability relation. Formally this construction is achieved by defining a quantic nucleus on the quantale $q(N)$.

Definition 5.3.1
Let Q be a quantale. A function $j : Q \to Q$ is a *quantic nucleus* iff

$$\begin{aligned} a \leq b \Rightarrow j(a) \leq j(b) &\qquad \text{j is monotonic} \\ a \leq j(a) &\qquad \text{j is increasing} \\ j(a) = j(j(a)) &\qquad \text{j is idempotent} \end{aligned}$$

and

$$j(a) \otimes j(b) \leq j(a \otimes b) \quad \text{for all } a, b \in Q \ .$$

□

The first three conditions state that j is a *closure operator*. The final condition makes j a looser notion than a morphism, because it is not required to preserve the monoidal operation.

The crucial element of the construction is the fact that the reachability relation gives rise to a quantic nucleus. First of all we need to define a new ordering relation on markings which then also extends to sets of markings.

Definition 5.3.2
Let $m_1, m_2 \in mark(N)$. Define

$$m_1 \leq m_2 \quad \text{iff } m_1 \text{ is reachable from } m_2 \ .$$

Let M_1, M_2 be sets of markings. Then $M_1 \leq M_2$ iff for every $m_1 \in M_1$ there exists $m_2 \in M_2$ such that $m_1 \leq m_2$. □

Given an ordering we can always define a downward closure operator. We shall call this operator *forward evolution*.

Definition 5.3.3
Let N be a net. For $A \subseteq mark(N)$ the *forward evolution* of A is the endofunction:

$$\downarrow(A) = \{m | \exists a \in A : m \leq a\} \ .$$

□

The meaning of the function $\downarrow$ is that it maps a set of markings M to the set of markings that are reachable from the markings in M.

It is obvious that

$$A \leq B \quad \text{iff} \quad A \subseteq B$$

when A, B are downward closed.

This gives us our quantic nucleus.

Proposition 5.3.4
$\downarrow : q(N) \to q(N)$ is a quantic nucleus. □

Now proving that closure under reachability defines a quantic nucleus is really not enough, because a quantic nucleus on a quantale Q is not necessarily a quantale endomorphism. But the following theorem by Niefield and Rosenthal (1988) tells us that a quantic nucleus is a quantale morphism, although not an endomorphism.

Proposition 5.3.5
Let Q be a quantale and $j : Q \to Q$ a quantic nucleus. Then the image of j is a quantale Q_j with $a \otimes_j b = j(a \otimes b)$ and $j : Q \to Q_j$ is a quantale morphism. □

So our wanted quantale is the image of the operation $\downarrow(q(N))$.

Proposition 5.3.6
The tuple $\langle \downarrow(q(N)), \subseteq, \otimes, \downarrow(\underline{0}) \rangle$ where the operation $M_1 \otimes M_2$ is defined as $\downarrow(M_1 + M_2)$ is a quantale. □

The elements of the quantale are sets of markings closed under forward evolution and the ordering on the sets is subset inclusion. Engberg and Winskel (1994) take the dual interpretation. They define the ordering $\leq'$ on markings by

$$m \leq' m' \text{ iff } m \to m' \ .$$

Then they also take downward closed subsets of markings. It is easy to see, that this corresponds to taking upward closed subsets with the ordering of the previous section, that is

$$\uparrow(A) = \{m | \exists a \in A : a \leq m\} \ .$$

It is routine to verify that $\uparrow$ defines a quantic nucleus.

What is the difference between these two interpretations? The main difference is in the intuitive interpretation of the connectives. A proposition is interpreted by its possible gain in the case of forward reachability and as the sufficient requirement in the case of backward reachability. In terms of linear logic formulas the difference is best seen with the way we state that b is reachable from a. In Engberg's and Winskel's approach the statement is expressed by the formula $a \multimap b$. As presented here the statement is expressed by the formula $b \multimap a$.

The rest of this section is concerned with the proof of the functoriality of $\downarrow$. The result is intuitively clear since a net morphism preserves the reachability and thus should in some way be compatible with the closure operators. The main question then is that of the definition of $\downarrow$ for a net morphism. The obvious extension of f to quantales does not work, because:

Lemma 5.3.7
Let $f : N_1 \to N_2$ be a <u>PetriG</u> morphism. Then generally *not* $f(M) = \downarrow^{N_2}(f(M))$, where M is a set of markings of N_1 and $\downarrow^{N_2}$ is the quantic nucleus generated by the reachability relation of N_2. □

This is so, because if for example the net N_1 is a subnet of N_2 then clearly the set of markings reachable from m in N_1 is smaller than the set of markings reachable from m in N_2. But on the other hand it would be reasonable to conjecture that by mapping a closed set of markings in N_1 to the closure of the corresponding markings in N_2 we would get a functor. Indeed this is the case as the following theorem shows.

Theorem 5.3.8
Let $f : N_1 \to N_2$ be a <u>PetriG</u> morphism, and let $\downarrow q(N_1), \downarrow q(N_2)$ be the corresponding net-quantales. Then the assignment $\downarrow(f)(M) = \downarrow^{N_2}(f(M))$ for $M \in \downarrow q(N_1)$ defines a functor <u>PetriG</u> $\to$ <u>Quant</u>.

Proof:
Essentially we have to prove, that $\downarrow(f) : \downarrow q(N_1) \to \downarrow q(N_2)$ is a quantale morphism. In the sequel let $A, B \in \downarrow q(N_1)$.

1.

$$\begin{array}{rcl} & & \downarrow(f) \text{ is monotonic:} \\ A \leq B & \Rightarrow & A \subseteq B \\ & \Rightarrow & f(A) \subseteq f(B) \\ & \Rightarrow & \overset{N_2}{\downarrow}(f(A)) \subseteq \overset{N_2}{\downarrow}(f(B)) \\ & \Rightarrow & \overset{N_2}{\downarrow}(f(A)) \leq \overset{N_2}{\downarrow}(f(B)) \\ & \Rightarrow & \downarrow(f)(A) \leq \downarrow(f)(B)\,. \end{array}$$

2.

$$\begin{array}{rcl} & & \downarrow(f) \text{ preserves joins:} \\ \downarrow(f)(A \vee B) & = & \overset{N_2}{\downarrow}(f(A \vee B)) \\ & = & \overset{N_2}{\downarrow}(f(A) \vee f(B)) \\ & = & \overset{N_2}{\downarrow}(f(A)) \vee \overset{N_2}{\downarrow}(f(B)) \\ & = & \downarrow(f)(A) \vee \downarrow(f)(B)\ . \end{array}$$

3.

$$\begin{array}{rcl} & & \downarrow(f) \text{ preserves the tensor:} \\ \downarrow(f)(A \otimes_{N_1} B) & = & \downarrow(f)(\overset{N_1}{\downarrow}(A + B)) \\ & = & \overset{N_2}{\downarrow}(f(\overset{N_1}{\downarrow}(A + B))) \\ & \subseteq & \overset{N_2}{\downarrow}(\overset{N_2}{\downarrow}(f(A + B))) \\ & = & \overset{N_2}{\downarrow}(f(A + B)) \\ & = & \overset{N_2}{\downarrow}(f(A) + f(B))) \\ & = & \overset{N_2}{\downarrow}(f(\overset{N_1}{\downarrow}(A)) + f(\overset{N_1}{\downarrow}(B))) \\ & \subseteq & \overset{N_2}{\downarrow}(f(\overset{N_1}{\downarrow}(A + B))) \\ & = & \downarrow(f)(A \otimes_{N_1} B)\ . \end{array}$$

Thus $\downarrow(f)(A \otimes_{N_1} B) = \downarrow^{N_2}(f(A) + f(B)))$.

$$\begin{array}{rcl} \overset{N_2}{\downarrow}(f(A) + f(B))) & = & \overset{N_2}{\downarrow}(\overset{N_2}{\downarrow}(f(A)) + \overset{N_2}{\downarrow}(f(B))) \\ & = & \overset{N_2}{\downarrow}(f(A)) \otimes_{N_2} \overset{N_2}{\downarrow}(f(B))) \\ & = & \downarrow(f)(A) \otimes_{N_2} \downarrow(f)(B)\ . \end{array}$$

In other words $\downarrow(f)(A \otimes_{N_1} B) = \downarrow(f)(A) \otimes_{N_2} \downarrow(f)(B)$.

4. $\downarrow(f)$ preserves the unit:

$$\downarrow(f)(\mathbf{I}_{N_1}) = \overset{N_2}{\downarrow}(f(\overset{N_1}{\downarrow}(\underline{0}))) \subseteq \overset{N_2}{\downarrow}(\overset{N_2}{\downarrow}(f(\underline{0}))) = \overset{N_2}{\downarrow}(\underline{0}) = \mathbf{I}_{N_2}\ .$$

But clearly $\underline{0} \subseteq \downarrow^{N_1}(\underline{0})$ which implies $f(\underline{0}) \subseteq f(\downarrow^{N_1}(\underline{0}))$, so that $\mathbf{I}_{N_2} \subseteq \downarrow^{N_2}(f(\downarrow^{N_1}(\underline{0}))) = \downarrow(f)(\mathbf{I}_{N_1})$. Thus

$$\downarrow(f)(\mathbf{I}_{N_1}) = \mathbf{I}_{N_2}\ .$$

□

5.4 Algebraic nets and linear logic

In this subsection we will show how linear logic can be used to express properties of algebraic nets. To formulate properties of an algebraic net in a language L_Ω we must define how the algebraic net generates a language.

Definition 5.4.1
The first order signature Ω_{AN} generated by an algebraic net AN is given by

1. $S = S_{AN}$,
2. $\Sigma = \Sigma_{AN}$, and
3. $\Pi = P$ where P is the set of places of the net.

The language generated by Ω_{AN} will be denoted by L_{AN}. □

Notice that all the predicates in our language will be monadic predicates.

Recall that the composition $\downarrow(\mathsf{Unf}(A))$ defines a quantale Q_{AN} for the net AN. The language L_{AN} can now be interpreted in the quantale Q_{AN}.

Definition 5.4.2
The interpretation of the language L_{AN} in the quantale Q_{AN} is given by the following rules. For reasons of clarity we have omitted explicit mention of the quantale in the interpretation.

1. $[\![\pi(t)]\!] = \begin{cases} \downarrow(\langle\pi, \mathrm{eval}_A(t)\rangle) & \text{iff } t \text{ is a ground term} \\ \downarrow(\bigcup_{\mathrm{ass}_A \in [Var(t)\to A]}(\langle\pi, \mathrm{ass}^{\#}_A(t)\rangle)) & \end{cases}$
2. $[\![1]\!] = \{$the set of all reachable markings of the net$\}$,
3. $[\![0]\!] = \varnothing$,
4. $[\![\mathbf{I}]\!] = \downarrow(\underline{0})$,
5. $[\![A \otimes B]\!] = \downarrow\{a + b | a \in [\![A]\!] \text{ and } b \in [\![B]\!]\}$,
6. $[\![A \& B]\!] = [\![A]\!] \bigcap [\![B]\!]$,
7. $[\![A \oplus B]\!] = [\![A]\!] \bigcup [\![B]\!]$,
8. $[\![A \multimap B]\!] = \bigcup\{[\![C]\!] | [\![C \otimes A]\!] \subseteq [\![B]\!]\}$,
9. $[\![(\bigvee x^s)P]\!] = \bigcap_{c\in\Sigma} [\![P[x/c]]\!]$, where c is of type $c :\to s$, and

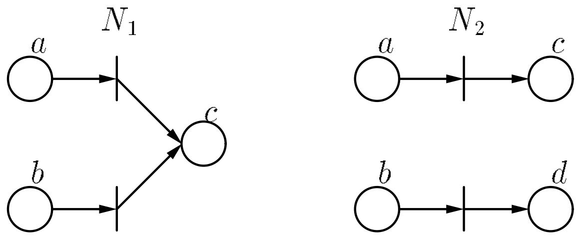


Figure 5.2: Nets showing the difference between the two choice operators.

10. $[\![(\bigwedge x^s)P]\!] = \bigcup_{c \in \Sigma} [\![P[x/c]]\!]$ with c of type $c :\rightarrow s$.

□

The idea behind this interpretation is the following. A predicate describes a place. The denotation of the predicate π is the set of all markings obtainable from the possible markings of the corresponding place, in other words the set of resources we could obtain if we had π. The other connectives and constants can now be understood as follows.

The constants $1, 0$ are the top and bottom of the quantale respectively. The constant $\mathbf{I}$ is the set of markings reachable from the empty marking. The interpretation of $[\![A \otimes B]\!]$ expresses the fact that to obtain some marking $m \in [\![A \otimes B]\!]$, we need resources $a \in [\![A]\!]$ and $b \in [\![B]\!]$ simultaneously. The additives are interpreted as choice operators. If we have a consequence m of $A \& B$, we know that regardless of our choice of A or B m will always be a consequence of A and B. On the other hand if m is a consequence of $A \oplus B$ it is obtained a non-deterministic choice between resources A and B. Thus it must be a consequence of either A or B. The nets in figure 5.4 show the difference between the two connectives. For net N_1 both $c \in [\![a \oplus b]\!]$ and $c \in [\![a \& b]\!]$ because we can obtain c from either a or b, while for N_2 only $c \in [\![a \oplus b]\!]$ because c can only be obtained from a.

The interpretation of $A \multimap B$ expresses the idea, that no consequence of $A \multimap B$ can give us more when taken together with some c than we would gain by having the appropriate $b \in [\![B]\!]$. This gives us the nice lemma

Lemma 5.4.3
$\models m \multimap m'$ iff marking m is reachable from m'. □

The interpretation of $(\bigvee x)P$ is the set of markings that are reachable regardless of the identity of the specific markings of P. That is each marking $m \in [\![(\bigvee x)P]\!]$ must be a consequence of $[\![\mathrm{ass}_A^{\#}|_{\{x\}}(P)]\!]$ for any legal assignment $\mathrm{ass}_A^{\#}|_{\{x\}}$. The interpretation of the $\bigwedge$-quantifier (some) is analogous.

Semantic entailment in the quantale is defined as

$$A_1 \otimes \ldots \otimes A_n \models A \ \text{ iff } \ [\![A_1]\!] \otimes \ldots \otimes [\![A_n]\!] \subseteq [\![A]\!] \, .$$

The truth of an Ω-sentence with respect to a quantale Q_{AN} is defined in terms of the ordering relation and the interpretation of the logical constant $\mathbf{I}$.

Definition 5.4.4
An Ω-sentence P is true in a quantale Q_{AN} (i.e. $Q_{AN} \models P$)

$$\text{iff } \ [\![\mathbf{I}]\!] \subseteq [\![P]\!] \, .$$

□

We shall also say that the property P holds in a net AN iff $[\![\mathbf{I}]\!] \leq_{Q_{AN}} [\![P]\!]$.

Proposition 5.4.5
The interpretation as given above is sound with respect to the sequent calculus,

$$A_1, \ldots, A_n \vdash A \Rightarrow A_1 \otimes \ldots \otimes A_n \models A \, .$$

□

Some examples will illustrate the definitions. Let AN be the dining philosophers net of figure 2.1 on page 10 with the interpretation of figure 2.4. We omit some of the more tedious calculations.

Example 5.4.6
Philosopher 1 can eat if he can get both forks:

$$Q_{AN} \models e(ph_1) \multimap p(ph_1) \otimes f(g_1) \otimes f(g_2),$$

because

$$[\![p(ph_1) \otimes f(g_1) \otimes f(g_2)]\!] = \{\langle p, ph_1\rangle + \langle f, g_1\rangle + \langle f, g_2\rangle, \langle e, ph_1\rangle\},$$

and

$$[\![e(ph_1)]\!] = \{\langle p, ph_1\rangle + \langle f, g_1\rangle + \langle f, g_2\rangle, \langle e, ph_1\rangle\}$$

clearly imply that

$$\underline{0} \in \{[\![C]\!] | [\![C \otimes e(ph_1)]\!] \subseteq [\![p(ph_1) \otimes f(g_1) \otimes f(g_2)]\!]\} \, .$$

□

Example 5.4.7
If philosophers ph_1 and ph_2 are hungry and there are enough forks it is possible for exactly one philosopher to start eating:

$$Q_{AN} \models e(ph_1) \oplus e(ph_2) \multimap p(ph_1) \otimes p(ph_2) \otimes f(g_1) \otimes f(g_2) \otimes f(g_3),$$

because

$$\begin{aligned} A = [\![e(ph_1) \oplus e(ph_2)]\!] &= [\![e(ph_1)]\!] \cup [\![e(ph_2)]\!] = \\ & \{\langle e, ph_1\rangle, \\ & \langle e, ph_2\rangle, \\ & \langle p, ph_1\rangle + \langle f, g_1\rangle + \langle f, g_2\rangle, \\ & \langle p, ph_2\rangle + \langle f, g_2\rangle + \langle f, g_3\rangle\}, \end{aligned}$$

while

$$\begin{aligned} & B = [\![p(ph_1) \otimes p(ph_2) \otimes f(g_1) \otimes f(g_2) \otimes f(g_3)]\!] = \\ & \qquad \{\langle p, ph_1\rangle + \langle p, ph_2\rangle + \langle f, g_1\rangle + \langle f, g_2\rangle + \langle f, g_3\rangle, \langle e, ph_1\rangle, \langle e, ph_2\rangle\}, \end{aligned}$$

so that

$$\{[\![C]\!] | [\![C \otimes A]\!] \subseteq B\} = \{\underline{0}, \langle f, g_2\rangle, \langle f, g_1\rangle\}$$

and thus

$$\underline{0} \in [\![e(ph_1) \oplus e(ph_2) \multimap p(ph_1) \otimes p(ph_2) \otimes f(g_1) \otimes f(g_2) \otimes f(g_3)]\!] \,.$$

$\square$

Example 5.4.8
But

$$Q_{AN} \not\models e(ph_1) \& e(ph_2) \multimap p(ph_1) \otimes p(ph_2) \otimes f(g_1) \otimes f(g_2) \otimes f(g_3),$$

because

$$[\![e(ph_1) \& e(ph_2)]\!] = [\![e(ph_1)]\!] \cap [\![e(ph_2)]\!] = \varnothing$$

and thus

$$\underline{0} \notin [\![e(ph_1) \& e(ph_2) \multimap p(ph_1) \otimes p(ph_2) \otimes f(g_1) \otimes f(g_2) \otimes f(g_3)]\!] \,.$$

What this means is that it is not possible for two philosophers to be eating at the same time in our model if we only have 3 forks. $\square$

Example 5.4.9
For any philosopher there is some fork that is not needed for the philosopher to start eating:

$$Q_{AN} \models \bigvee x^{phil} \,.\, \bigwedge y^{fork} \,.\, e(x) \otimes f(y) \multimap p(x) \otimes f(g_1) \otimes f(g_2) \otimes f(g_3) \,.$$

The interpretation of this formula is complicated and we shall only sketch the details.

$$\begin{aligned}&[\![\bigvee x^{phil} \,.\, \bigwedge y^{fork} \,.\, e(x) \otimes f(y) \multimap p(x) \otimes f(g_1) \otimes f(g_2) \otimes f(g_3)]\!] = \\ &\quad \bigcup_{ph_i \in phil} \Big(\bigcap_{f_i \in fork} [\![e(ph_i) \otimes f(f_i) \multimap p(ph_i) \otimes f(g_1) \otimes f(g_2) \otimes f(g_3)]\!]\Big)\end{aligned}$$

Now,

$$\begin{aligned}&[\![p(ph_i) \otimes f(g_1) \otimes f(g_2) \otimes f(g_3)]\!] = \\ &\quad \{\langle p, ph_i\rangle + \langle f, g_1\rangle + \langle f, g_2\rangle + \langle f, g_3\rangle, \langle e, ph_i\rangle + \langle f, g_{i+2 \bmod 3}\rangle\},\end{aligned}$$

and

$$\begin{aligned}&[\![e(ph_i) \otimes f(g_j)]\!] = \\ &\quad \{\langle p, ph_i\rangle + \langle f, g_i\rangle + \langle f, g_{i+1 \bmod 3}\rangle + \langle f, g_j\rangle, \langle e, ph_i\rangle + \langle f, g_j\rangle\} \,.\end{aligned}$$

One can now see, that

$$\underline{0} \in [\![e(ph_i) \otimes f(g_j) \multimap p(ph_i) \otimes f(g_1) \otimes f(g_2) \otimes f(g_3)]\!]$$

for the pairs $(ph_1, f_3), (ph_2, f_1), (ph_3, f_2)$ and this then leads to the result that

$$\underline{0} \in [\![\bigvee x^{phil} \,.\, \bigwedge y^{fork} \,.\, e(x) \otimes f(y) \multimap p(x) \otimes f(g_1) \otimes f(g_2) \otimes f(g_3)]\!] \,.$$

□

Example 5.4.10
The following two formulas describe transitions t_1 and t_2 respectively:

$$\begin{aligned} Q_{AN} &\models \bigvee x^{phil} \,.\, e(x) \multimap p(x) \otimes f(l(x)) \otimes f(r(x)), \\ Q_{AN} &\models \bigvee x^{phil} \,.\, p(x) \otimes f(l(x)) \otimes f(r(x)) \multimap e(x) \,.\end{aligned}$$

because

$$\begin{aligned}&[\![p(x) \otimes f(l(x)) \otimes f(r(x))]\!] = \\ &\quad \bigcup_{i \in \{1,2,3\}} \{\langle p, ph_i\rangle + \langle f, g_i\rangle + \langle f, g_{i+1 \bmod 3}\rangle, \langle e, ph_i\rangle\},\end{aligned}$$

and thus

$$\underline{0} \in [\![\bigvee x^{phil} \,.\, e(x) \multimap p(x) \otimes f(l(x)) \otimes f(r(x))]\!] \,.$$

Analogously for t_2. □

Notice also the form of the lhs of the implication in example 5.4.9. We had to explicitly mention the left over fork, although what we would have wanted to be able to say something like "from the initial marking any philosopher can start eating". For this end we need a "wildcard" marking.

Lemma 5.4.11
$[\![m \otimes 1]\!] = \downarrow\{m' | m' \leq m\}$.

Proof:

$$\begin{aligned} [\![m \otimes 1]\!] &= \downarrow\{m + b | b \in [\![1]\!]\} \\ &= \downarrow\{m' | m' \leq m\} \,. \end{aligned}$$

□

Example 5.4.9 now becomes:

$$Q_{AN} \models \bigvee x^{phil} \,.\, e(x) \otimes 1 \multimap p(x) \otimes f(g_1) \otimes f(g_2) \otimes f(g_3) \,.$$

In this chapter we have shown how to extend a correspondence between linear logic and Petri nets to a correspondence between intuitionistic predicate linear logic and algebraic high-level nets. The examples presented suggest that linear intuitionistic predicate logic could be used as a foundation for a logic to reason about high-level nets. As such the logic is not very expressive, because it is not possible to reason about transitions, nor is it possible to express negative facts about the net. However Engberg and Winskel (1994) have recently discovered that in so called atomic nets (a net is atomic iff whenever $M \to \underline{0}$ then $\underline{0} \to M$) we have that $\models A \& \mathbf{I} \multimap 0$ iff $\not\models A$, thus making it possible to assert the non-reachability of markings. Since most nets occurring in practice are atomic, it seems useful to further study the expressiveness of linear logic. On the other hand linear logic also contains features whose importance for reasoning about nets is not yet clear, eg. as pointed out by C. Brown (1991), the interpretation of the choice operators is problematic, because they can be thought of as internal and external choice (Marti-Oliet and Meseguer 1989). But this distinction is not very sensible in Net theory because one makes the assumption that no external observer can induce a transition to fire. Also an interesting observation one can make is that that linear predicates seem to correspond to the dynamic predicates used in the definition of Pr/T-nets (Genrich 1986).

Chapter 6

Conclusions

One of the aims of this thesis was to try to answer the question "What is a high-level net?". We have tried to answer the question from several different angles. However since we actually have worked only with Algebraic nets, we also have to answer the question to what extent we really have said anything about high-level nets in general. What we feel we have done, is tried to choose a high-level net formalism as simple as possible, so that we can concentrate on the basic common aspects of all known high-level classes, namely the unfolding semantics, the idea of "symbolic firing" through substitution, and the motivation of the nets through folding.

In chapter 2 we studied the unfolding construction of Algebraic nets. However at each stage we tried to identify those components of the construction that were not dependent on the specification formalism used in the annotations. As we then showed the only component in the unfolding construction that needed to be redefined was the notion of a *consistent transition assignment*. For both Algebraic nets and Order-sorted Algebraic nets it is simply the set of all assignments to the variables of the transition, while for Algebraic nets with conditions it is the set of those assignments that satisfy the conditions on the transition. An interesting avenue for further research would be to try and abstract the notion of consistent transition assignment to see whether other high-level like net-classes, like timed-nets or stochastic nets could be made to fit into this framework.

In section 3.1.2 we described a high-level net as a graph in the category of monoids over a substitution system, thus formalizing the idea of "symbolic firing". Although we have again used Algebraic nets as our example formalism, we were able to describe the construction of an Algebraic net in terms of a tensor product of the sketch of a substitution systems, the sketch of graphs and the sketch of commutative monoids. By suitably changing the substitution system (in the case of Algebraic nets it is many-sorted algebra) we obtain other high-level net formalisms. This abstraction also suggests that some new formalisms, that cannot quite be considered high-level Petri nets, are obtained by changing either the sketch of graphs or the sketch of commutative monoids. By replacing the sketch of monoids with some axiomatization of a process algebra we could obtain a categorical semantics of this process algebra with value passing. However it is yet not quite clear what a suitable replacement for the sketch of graphs would be so that the formalism would still retain its net-like character.

In chapter 4 we described a high-level net as a morphism between two Place/-Transition nets. Although the construction is only given for Algebraic nets we can divide it into two parts, one generic and the other specific to Algebraic nets. The way the splitting of the annotations and the signature of the Algebraic net are calculated is generic. The same principle can be used with any other formalism that uses a similar notion of morphism. But on the other hand the coding of the folding into the equations is naturally very specific to many-sorted algebra. However it should be noted that the construction as given here codes everything into annotation functions and that the resulting annotation functions are surjective. Thus it is not possible to construct conditions on transitions or annotation functions that are defined only on a subset of the domain.

Finally in chapter 5 we described high-level nets as set of formulae of linear logic. Although we only gave an encoding of Algebraic nets as formulae of linear logic it is clear that both Order-sorted nets and Algebraic nets with conditions can also be translated into formulae of linear logic. This can be done by either first translating the nets into Algebraic nets or by extending the encoding to accommodate subsorts or the conditions. Subsorts are added by simply changing the definition of the language, while conditions can be incorporated as extra predicates in the antecedent of the linear implication that describes the transition. However the exact details of these constructions are left for further research.

Let us now look at the two other prominent net-classes, Pr/T-nets and Coloured nets, and our reasons for not choosing them as the starting point of our investigation. Pr/T-nets combine first order logic with C/E-systems in their basic form. Pr/T-nets have been extended to multi-sets of tokens, this extension however has been done by adding "tags" to the markings, ie. $2x$ in our formalism would become $\langle x, 0\rangle + \langle x, 1\rangle$ in a Pr/T-net. The formalism of Pr/T-nets also contains several features like conditions on transitions and conditional sums, that make the modeling of a systems easier. The main reason for not choosing to work with Pr/T-nets was the treatment of multisets. Since we wanted to extend the Petri Nets are Monoids approach we needed a high-level net class that was directly built on P/T-nets. Coloured Nets do use multisets of tokens in a natural way. However for Coloured Nets "the set of allowable expressions and predicates is not explicitly given"(Jensen 1986, p. 250). For example the Coloured Nets in the Design/CPN tool use a version of Standard ML (Milner, Tofte and Harper 1990) as the annotation language. So we feel that the advantages of Algebraic nets are that it is based on ideas of the Petri Nets are Monoids approach and that there exists a well understood theory of substitutions in many-sorted algebra, so that we have a very natural mathematical framework in which to do our investigation. However given suitable formalizations or translations of the annotation formalisms of both Pr/T-nets and Coloured Nets the results presented in this thesis should be applicable to these net-classes.

Let us finally briefly look at the possibilities of exploiting the results presented in this work for the development of tools. The unfolding semantics described in chapter 2 is directly applicable to the development of analysis tools. This idea has already been implemented in the tool `PROD` (Grönberg, Tiusanen and Varpaaniemi 1993). Clearly the idea of "symbolic firing" described in section 3.1.2 can also be used to build an analyzer for high-level nets. However it is more efficient to use the unfolding semantics because the test for the enabledness of a transition involves the calculation of unifiers. As a transition can be enabled many times by the same transition assignment these calculations can become very costly. By first unfolding the net the unifiers are calculated only once. On the other hand the "abstract" arrows present in the structured transition system semantics, that were hinted at at the end of section 3.1.2, should be investigated further. They might provide some possibilities for more efficient reachability analysis. As shown in section 4.3 the folding construction described in section 4.2 can be used to calculate deadlock preserving skeletons. However to evaluate the practicality of this method the complexity of calculating the relation $\leftrightarrow$ should be examined. Finally in chapter 5 we describe an axiomatization of high-level nets. From the point of view of tools, this axiomatization is not very promising. However, the use of linear logic as a query language for an reachability analysis tool should be explored further.

Appendix

The appendix contains short reviews of category theory, the Petri Nets are monoids approach, and universal algebra. The aim is to give the basic definitions and fix the notation. A tutorial type of presentation is not aimed at.

A.1 Basic category theory

The basic idea underlying category theory is that the crucial mathematical properties of a given subject do not reside within the *structures* in question, and even less in the particular *representation* chosen for them, but rather in the mappings that preserve those structures. Thus category theory emphasizes mappings before objects. Indeed most of category theory could be written without direct reference to objects.

Another equivalent way to characterize category theory is as a "diagrammatic language of arrows". In this language mappings are represented by arrows (a mapping $f : a \to b$ is $a \xrightarrow{f} b$). Most theorems then are theorems that state equality of arrows under certain conditions. These are represented by commutativity diagrams. Eg. if we require that $f; g = h; j$ with $f : a \to b$, $g : b \to c$, $h : a \to b'$, $j : b' \to c$ we state that the diagram below

$$\begin{tikzcd} a \arrow[r, "f"] \arrow[d, "h"'] & b \arrow[d, "g"] \\ b' \arrow[r, "j"'] & c \end{tikzcd}$$

commutes.

In this appendix we give a short introduction to the basic concepts in category theory. The main emphasis is on the intuition behind the definitions. Most examples will be with sets and functions, but some knowledge of algebra is required. This appendix only discusses concepts relevant to this work.

The appendix is structured as follows. We first define the notions of category, functor and natural transformation. The we discuss limits and co-limits, which are ways of expressing combinations of objects. Finally we define the notion of adjunction. For missing proofs we refer to (MacLane 1971).

A.1.1 Basic category theory

The aim of category theory is to study the mappings between mathematical objects. We start this introduction by defining a category.

Definition A.1.1
A category $\underline{\mathsf{C}}$ is a collection of objects $|\underline{\mathsf{C}}|$ such that

- For each pair of objects (a, b) of $\underline{\mathsf{C}}$ a set $\mathbf{Mor}_{\underline{\mathsf{C}}}(a, b)$ called the *set of morphisms from a to b*, with $\mathbf{Mor}_{\underline{\mathsf{C}}}(a, b)$ and $\mathbf{Mor}_{\underline{\mathsf{C}}}(a', b')$ disjoint unless $a = a'$ and $b = b'$ in which case they coincide. We shall take the naive view that all sets are "proper" sets. We will not encounter any foundational problems with this.
- For any three objects a, b, c of $|\underline{\mathsf{C}}|$ there is a mapping (composition)

  $$\mathbf{Mor}_{\underline{\mathsf{C}}}(a, b) \times \mathbf{Mor}_{\underline{\mathsf{C}}}(b, c) \to \mathbf{Mor}_{\underline{\mathsf{C}}}(a, c)$$

  described by $(f, g) \mapsto f; g$, (notice that the order of the composition is written in the order of the arrows), with the following properties:

  - For each object a there is a morphism $\mathbf{id}_a \in \mathbf{Mor}_{\underline{\mathsf{C}}}(a, a)$ which is right identity under ; for the elements of $\mathbf{Mor}_{\underline{\mathsf{C}}}(a, b)$ and left identity under ; for the elements $\mathbf{Mor}_{\underline{\mathsf{C}}}(b, a)$.
  - ; is associative in the sense that when the composites $f; (g; h)$ and $(f; g); h$ are defined they are equal.

We will often use the notation $f : a \to b$ and $a \xrightarrow{f} b$ for the morphism sets and call them arrows. The use of small letters for both objects and arrows is to emphasize the fact that due to the identity morphisms objects can be manipulated as arrows. Thus ordinary function application $f(x)$ can and will sometimes be written $x; f$. □

To make the above abstract definition clear and to convince the reader about its generality, we give some examples

Example A.1.2
(The category of sets) The category $\underline{\mathsf{Set}}$ of sets and their mappings. The objects are ordinary sets and the morphisms are ordinary mappings between sets. Composition is the usual composition of mappings. $\square$

Example A.1.3
(Preorder) Let $(E, \preceq)$ be a preordered set. We can view this preordered set as a category $\underline{\mathsf{E}}$ as follows: take as objects of $\underline{\mathsf{E}}$ the elements of E and for $a, b \in E$ define

$$\mathbf{Mor}_{\underline{\mathsf{E}}}(a,b) = \begin{cases} \{(a,b)\} & \text{if } a \preceq b \\ \phi & \text{otherwise.} \end{cases}$$

Composition is defined by the transitivity of $\preceq$ and the identity morphisms by the reflexivity of $\preceq$. $\square$

Example A.1.4
(Monoid) A monoid is a set X equipped with a function $\otimes : X \times X \to X$ (monoid multiplication) and a distinguished element $\mathbf{I}$ (monoid identity) subject to the two laws:

$$\begin{aligned} x \otimes (y \otimes z) &= (x \otimes y) \otimes z \ \text{ for all } x, y, z \\ x \otimes \mathbf{I} = \quad x &= \mathbf{I} \otimes x \ \text{ for all } x. \end{aligned}$$

Now a monoid is a category with one object. To see this call the object A, then let $X = \mathbf{Mor}(A, A)$ where $\otimes$ is composition and $\mathbf{I}$ is the identity morphism. $\square$

Monoids and preorders viewed as categories are at opposite extremes. A monoid has one object and many morphisms, while a preorder has at most one morphism between objects.

Category theory tries to express every mathematical statement as a statement about arrows. This has the advantage, that a proof of the statement also gives a proof of the statement obtained by reversing the arrows. Usually we will "dualise" every statement and definition immediately. We do this for the definition of a category.

Definition A.1.5
The opposite category $\underline{C}^{op}$ is formed by turning around all the arrows in $\underline{\mathsf{C}}$. $\square$

As we are interested in studying structure preserving mappings between objects it seems natural to define mappings between categories. These are called functors.

Definition A.1.6
(Functor) A (covariant) functor from a category $\underline{\mathsf{A}}$ to a category $\underline{\mathsf{B}}$ is a pair of mappings that assign to every object $a \in |\underline{\mathsf{A}}|$ an object $F(a) \in |\underline{\mathsf{B}}|$ and to every morphism $f : a \to b \in \mathbf{Mor}_{\underline{\mathsf{A}}}$ a morphism $F(f) : F(a) \to F(b) \in \mathbf{Mor}_{\underline{\mathsf{B}}}$ such that:

- $F(\mathbf{id}_a) = \mathbf{id}_{F(a)}$ for all $a \in |\underline{\mathsf{A}}|$
- if $f; g$ is defined in $\underline{\mathsf{A}}$ then $F(f); F(g)$ is defined in $\underline{\mathsf{B}}$ and $F(f); F(g) = F(f; g)$.

We use the notation $F : \underline{\mathsf{A}} \to \underline{\mathsf{B}}$ for a functor F from $\underline{\mathsf{A}}$ to $\underline{\mathsf{B}}$. A contravariant functor $F : \underline{\mathsf{A}} \to \underline{\mathsf{B}}$ is a covariant functor $F : \underline{\mathsf{A}} \to \underline{B}^{op}$. □

Thus a functor is a mapping that respects compositions and identities. Two special kinds of functors deserve mentioning. A bifunctor is a functor $F : \underline{\mathsf{A}} \times \underline{\mathsf{B}} \to \underline{\mathsf{C}}$. An endofunctor is a functor $F : \underline{\mathsf{A}} \to \underline{\mathsf{A}}$.

Example A.1.7
(Monoid homomorphisms) Let $\underline{\mathsf{M}}$ and $\underline{\mathsf{N}}$ be monoids viewed as categories. Then a monoid homomorphism $f : (M, \otimes_M, \mathbf{I}_M) \to (N, \otimes_N, \mathbf{I}_N)$ is a functor $H : \underline{\mathsf{M}} \to \underline{\mathsf{N}}$. □

Example A.1.8
(Monotonic functions) Let $(A, \preceq_A)$ and $(B, \preceq_B)$ be two preorders. Then a monotonic function $f : A \to B$ defines a functor as the reader easily can check. □

Having defined mappings between categories (i. e. functors), it is natural to ask whether we could also define mappings between functors. This is indeed so and these maps are called natural transformations.

Definition A.1.9
(Natural transformation) If $F, G : \underline{\mathsf{A}} \to \underline{\mathsf{B}}$ are functors then a natural transformation from F to G is a rule that assigns to each object $a \in |\underline{\mathsf{A}}|$ a morphism $\eta_a : F(a) \to G(a) \in \mathbf{Mor}_{\underline{\mathsf{B}}}$ in such a way that associated with every morphism $f : a \to b \in \mathbf{Mor}_{\underline{\mathsf{A}}}$ there is a commutative diagram:

$$
\begin{tikzcd}
F(a) \arrow[r, "\eta_a"] \arrow[d, "F(f)"'] & G(a) \arrow[d, "G(f)"] \\
F(b) \arrow[r, "\eta_b"'] & G(b)\,.
\end{tikzcd}
$$

□

Example A.1.10
A good example of a natural transformation is the evaluation of a function at an argument. Let B^A denote the set of all functions from set A to set B. Now define $eval : B^A \times A \to B$ as $eval(f, a) = f(a)$.

Fix a specific A. The map $B \mapsto B^A \times A$ extends to a functor $F : \underline{\mathsf{Set}} \to \underline{\mathsf{Set}}$. So for this specific A $eval : F \to I_{\underline{\mathsf{Set}}}$. This is equivalent to the following diagram:

$$\begin{tikzcd}
B^A \times A \arrow[r, "eval_a"] \arrow[d, "F(f)"'] & B \arrow[d, "I(f)"] \\
C^A \times A \arrow[r, "eval_b"'] & C\,.
\end{tikzcd}$$

So we see that $eval$ is a natural transformation. □

This process of defining mappings between objects and then taking these mappings as objects and defining new mappings between these could be continued "ad nauseum". In practice the usefulness of mappings between natural transformations (or higher order mappings) is very limited and we shall not encounter them in this work. Instead we turn to another important issue in category theory.

A.1.2 Limits

In the previous paragraphs we have discussed many different kinds of mappings as the basic ingredient of category theory. Category theory is also concerned with characterizing mathematical constructions in the arrow-theoretic language. As it turns out the categorical notion of *universality* is disguised in many mathematical constructions: equivalence relations, complete metric spaces, etc. The concept of a universal construction allows us to describe these constructions in an uniform manner as universal objects or universal arrows. Universal arrows are usually described by statements like "for every f there exists a unique f' such that $uf' = f$ ". The arrow u is then an universal arrow. We will here concern ourselves with a special kind of universal constructions, namely *limits* and their duals *co-limits*. Below when we use the word limit, we usually mean both limits *and* co-limits. Our first example of a limit is the categorical product, a generalization of a cartesian product of two sets.

Definition A.1.11
(Product) Let a, b be objects in $\underline{C}$. A product of a and b in $\underline{C}$ is an object $a \times b$ with two morphisms $\pi_1 : a \times b \to a, \pi_2 : a \times b \to b$, called the projections

(left and right), if for every other object c in $\underline{\mathrm{C}}$ and every pair of morphisms $\langle f, g\rangle$ with $f : c \to a, g : c \to b$ there exists a unique morphism $h : c \to a \times b$ such that the following diagram commutes:

$$\begin{tikzcd}
a & a \times b \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] & b \\
& c \arrow[ul, "f"] \arrow[u, "h"] \arrow[ur, "g"'] &
\end{tikzcd}$$

□

Again we dualise:

Definition A.1.12
(Coproduct) Let a, b be objects in $\underline{\mathrm{C}}$. A coproduct of a and b in $\underline{\mathrm{C}}$ is an object $a + b$ with two morphisms $\iota_1 : a \to a + b, \iota_2 : b \to a + b$, called the injections (left and right) if for every other object c in $\underline{\mathrm{C}}$ and every pair of morphisms (f, g) with $f : a \to c, g : b \to c$ there exists a unique morphism $h : a + b \to c$ such that the following diagram commutes:

$$\begin{tikzcd}
a \arrow[r, "\iota_1"] \arrow[dr, "f"'] & a + b \arrow[d, "h"] & b \arrow[l, "\iota_2"'] \arrow[dl, "g"] \\
& c &
\end{tikzcd}$$

□

Example A.1.13
In $\underline{\mathrm{Set}}$ a product is simply the cartesian product of two sets. A coproduct is the disjoint union of two sets. In a preorder viewed as a category products and coproducts correspond to join and meet respectively. □

A simpler kind of limit is obtained by constructing an empty coproduct. In $\underline{\mathrm{Set}}$ an empty coproduct consists of an empty set. The two injections are identical to the identity morphism. So the only interesting thing left is the universal arrow. Formally:

Definition A.1.14
(Initial object) An object $\top$ of $\underline{\mathrm{C}}$ is said to be an initial object if, for every other object x in $\underline{\mathrm{C}}$, there is only one arrow from $\top$ to x. □

Again dually we can ask what an empty product is and end up with the following:

Definition A.1.15
(Terminal object) An object $\perp$ in $\underline{C}$ is said to be a terminal object if for every other object x in $\underline{C}$ there is only one arrow from x to $\perp$. $\square$

Example A.1.16
In $\underline{\mathrm{Set}}$ the empty set ϕ is initial and $\{\phi\}$ or any other singleton set is terminal. Because of isomorphism it does not matter which singleton we choose. $\square$

Definition A.1.17
Given in $\underline{C}$ a pair of arrows $f, g : a \to b$ with the same domain a and codomain b, an equalizer of $\langle f, g\rangle$ is an arrow $u : e \to a$ (or, a pair $\langle e, u\rangle$) such that

- $u; f = u; g$,
- if $h : c \to a$ has $h; f = h; g$ then there exists an unique arrow $h' : c \to e$ such that $h'; u = h$. This is displayed in the commutativity requirement of the diagram below:

$$\begin{tikzcd}
e \arrow[r, "u"] & a \arrow[r, shift left, "f"] \arrow[r, shift right, "g"'] & b \\
c \arrow[u, "h'"] \arrow[ur, "h"] & &
\end{tikzcd}$$

$\square$

Definition A.1.18
Given in $\underline{C}$ a pair of arrows $f, g : a \to b$ with the same domain a and codomain b, a coequalizer of $\langle f, g\rangle$ is an arrow $u : b \to e$ (or, a pair $\langle e, u\rangle$) such that

- $f; u = g; u$,
- if $h : b \to c$ has $f; h = g; h$ then there exists an unique arrow $h' : e \to c$ such that $h = u; h'$. This is displayed in the commutativity requirement of the diagram below:

$$\begin{tikzcd}
a \arrow[r, shift left, "f"] \arrow[r, shift right, "g"'] & b \arrow[r, "u"] \arrow[dr, "h"'] & e \arrow[d, "h'"] \\
& & c
\end{tikzcd}$$

□

The coequalizer corresponds to an equivalence relation. In set-theoretical terms the coequalizer identifies those elements of b that are the image of the same x of a under the functions f and g.

Another important type of limit is the pullback.

Definition A.1.19
Given in $\underline{C}$ a pair of arrows $f : a \to c$ and $g : b \to c$ with the same codomain c, a pullback is given by an object $a \times_c b$ and arrows $\iota_1 : a \times_c b \to a$, $\iota_2 : a \times_c b \to b$ such that:

- $\iota_1; f = \iota_2; g$, and
- given any other object d and maps $k : d \to b, l : d \to a$ there exists a unique map $h : d' \to a \times_c b$ such that the following diagram commutes:

$$
\begin{tikzcd}
d \arrow[dr, "h"] \arrow[drr, "k"] \arrow[ddr, "l"'] & & \\
& a \times_c b \arrow[r, "\iota_2"] \arrow[d, "\iota_1"'] & b \arrow[d, "g"] \\
& a \arrow[r, "f"] & c
\end{tikzcd}
$$

□

The general notion of limit and its dual co-limit are based on the definition of a *cone* in a category, which is a special kind of a diagram.

Definition A.1.20
A *diagram* D in a category $\underline{C}$ is a graph homomorphism $D : I \to C$, where C is the underlying graph of the category. A *commutative cone* with vertex W over a diagram $D : I \to C$ is a natural transformation α from the constant functor with value W on I, to D, which implies that the following diagram

must commute for all $e : i \to j$:

$$\begin{tikzcd}
 & D(i) \arrow[dd, "D(e)"] \\
W \arrow[ur, "\alpha i"] \arrow[dr, "\alpha j"'] & \\
 & D(j)
\end{tikzcd}$$

□

Definition A.1.21
A commutative cone over a diagram D is called *universal* if every other commutative cone over the same diagram has a unique arrow to it. A universal cone, if it exists, is called the *limit* of the diagram D. □

A *co-limit* is a universal co-cone.

A.1.3 Adjunctions

Concepts that can be viewed as special cases of adjunctions were known long before the advent of category theory. One of the more classical examples is that of a *Galois connection* between two preordered sets.

Definition A.1.22
Let $\mathcal{A} \xrightarrow{f} \mathcal{B}$ and $\mathcal{B} \xrightarrow{g} \mathcal{A}$ be monotonic functions between two preorders $\mathcal{A} = \langle A, \preceq_A \rangle, \mathcal{B} = \langle B, \preceq_B \rangle$. The pair $\langle f, g \rangle$ is an *adjunction (Galois connection)* iff

1. f and g are monotonic and
2. the relations $f(a) \preceq_B b$ and $a \preceq_A g(b)$ are equivalent for all pairs of elements $(a, b) \in A \times B$.

□

By regarding a preorder $\mathcal{A}$ as a category $\underline{\mathrm{A}}$ with $|\mathbf{Mor}_{\underline{\mathrm{A}}}(a, b)| = 1$ (as in the example on page 105), when $a \preceq_A b$ (transitivity and reflexivity define composition and identities) and taking f and g as functors $F : \underline{\mathrm{A}} \to \underline{\mathrm{B}}$ and $G : \underline{\mathrm{B}} \to \underline{\mathrm{A}}$, the above definition transfers immediately into a category theoretical setting, allowing us to generalize the notion of a Galois connection to an adjunction as follows:

Definition A.1.23
An *adjunction* between categories $\underline{\mathrm{A}}$ and $\underline{\mathrm{B}}$ is a quadruple $(F, G, \eta, \varepsilon)$ where $F : \underline{\mathrm{A}} \to \underline{\mathrm{B}}$ and $G : \underline{\mathrm{B}} \to \underline{\mathrm{A}}$ are functors, the left and right adjoint respectively, and $\eta : 1_{\underline{\mathrm{A}}} \to FG$ and $\varepsilon : GF \to 1_{\underline{\mathrm{B}}}$ are natural transformations (unit and co-unit) such that:

$$\begin{aligned} (G\eta);(G\varepsilon) &= 1_G, \\ (F\eta);(F\varepsilon) &= 1_F \ . \end{aligned}$$

□

The most important property of an adjunction is expressed by the following fact.

Theorem A.1.24
Let $F : \underline{\mathrm{A}} \to \underline{\mathrm{B}}$ and $G : \underline{\mathrm{B}} \to \underline{\mathrm{A}}$ be functors such that F is left adjoint to G. Then F preserves co-limits and G preserves limits. □

A.2 Petri Nets are Monoids

In this appendix we will look at a category theoretic model of Petri nets, where the algebraic structure in which multisets are coded is that of a monoid, which has been proposed by Meseguer and Montanari (1990). It is based on the observation that the markings of the net obey a commutative monoidal law, where this monoidal structure is induced on the transitions by the firing rule.

The usefulness of viewing the multiset of places as a monoid will become clear in the following subsection. Mainly the monoidal structure on the places induces a monoidal structure on the transitions giving a rich hierarchy of categories with increasingly rich structures on the transitions. The monoidal structure on the transitions is used to represent the concurrent firing of transitions. Thus if we think of the monoidal operation on the places (conditions) as meaning the conditions hold simultaneously, the operation on the transitions has the reading the events happen simultaneously. Hence the monoidal operation can be used as a representation of parallelism, or in other words the algebraic structure of parallelism is monoidal.

In this model a very liberal view of nets is taken. For example in some constructions isolated places are useful. In some categories the net with only one transition and place connected in a loop plays a very important role by being the terminal net. Essentially this means that we wish to treat nets as graphs. Also the nets will not have initial markings. This means that the behavior of the net is a much more abstract concept. The behavior of the net includes *all* the possible behaviors of the net.

This appendix consists of two subsections. The first subsection discusses the different categories of nets that are derivable in the models, while the second subsection discusses the completeness properties of some specific categories in this model.

A.2.1 General structure of the model

In this subsection we shall first take a look at the different categories of nets. The categories are all derived from the category of graphs by adding structure to the set of transitions. Then we shall see that these categories are related by a sequence of adjunctions. Finally we shall try to give some characterizing examples so that the reader can get a feel for the differences between the morphisms in these categories.

The first thing we need to do is define a category of graphs (Meseguer and Montanari 1990).

Definition A.2.1
The category of graphs $\underline{\mathsf{Graph}}$ has as objects graphs ie. tuples $\langle T, P, \iota, o\rangle$, where T is the set of arcs, P is the set of nodes and ι, o are functions $T \to P$, and morphisms pairs $\langle f, g\rangle$ of functions such that the diagram

$$\begin{tikzcd}
T \arrow[r, shift left, "\iota"] \arrow[r, shift right, "o"'] \arrow[d, "f"'] & P \arrow[d, "g"] \\
T' \arrow[r, shift left, "\iota'"] \arrow[r, shift right, "o'"'] & P'
\end{tikzcd}$$

commutes. □

By adding a *free* monoidal structure on the places we get the category $\underline{\mathsf{Petri}}$.

Definition A.2.2
The category of *Petri nets* $\underline{\mathsf{Petri}}$ has as objects Petri nets $\langle T, P^{\otimes}, \iota, o\rangle$ and as morphisms graph morphisms $\langle f, g\rangle$ where g is a monoid homomorphism.

□

If we want *partial* maps on the transitions we can do it by adding a special element 0 to the set of transitions.

In this way we get the category of pointed Petri nets.

Definition A.2.3
The category of *pointed Petri nets* $\underline{\mathsf{Petri}}_0$ has as objects pointed nets $\langle (T,0), P^{\otimes}, \iota, o\rangle$ and as morphisms graph morphisms $\langle f, g\rangle$ where f is a pointed function and g is a monoid homomorphism. □

If we add a monoidal structure to the transitions we get the category of Petri commutative monoids.

Definition A.2.4
A *Petri commutative monoid* consists of a Petri net where the set of transitions is a commutative monoid $(T, +, 0)$ and where

$$\iota, o : (T, +, 0) \to P^{\otimes}$$

are monoid homomorphisms. A Petri commutative monoid homomorphism is a Petri net morphism $\langle f, g\rangle$ where f is monoid homomorphism. This defines a category $\underline{\mathsf{CMonPetri}}$. □

The reader should note that the monoid on the transitions need not be free. The idea here is that the monoidal structure on the transitions reflects the structure of the computation. If there are synchronization constraints in the system these are expressed as conditions on the transition monoid.

The above categories can all be augmented with reflexive structures. To each place we adjoin an *identity* transition that represents the fact that nothing happens at that specific place. The corresponding categories are $\underline{\mathsf{CMonRPetri}}$, $\underline{\mathsf{RPetri}}$, $\underline{\mathsf{RGraph}}$.

The last category in the hierarchy is the category of Petri categories.

Definition A.2.5
A *Petri category* is a small category $\underline{\mathsf{C}} = (P^{\otimes}, T, ;, id)$ whose set of objects is a free commutative monoid, and whose set of arrows has a commutative monoid structure $(T, \otimes, id_I)$, that is *not* necessarily free, and is compatible with the categorical structure in the sense that the source and target functions $\iota, o : T \to P^{\otimes}$ are monoid homomorphisms and that $\otimes$ respects identities and (sequential) composition:

1. $\iota(\alpha;\beta) = \iota(\alpha)$ and $o(\alpha;\beta) = o(\beta)$.
2. $\alpha; id(\iota(\alpha)) = \alpha$ and $id(o(\alpha)); \alpha = \alpha$.
3. $(\alpha;\beta);\gamma = \alpha;(\beta;\gamma)$.
4. Given $\alpha : u \to v, \alpha' : u' \to v', \beta : v \to w, \beta' : v' \to w'$, we have

$$(\alpha \otimes \alpha');(\beta \otimes \beta') = (\alpha;\beta) \otimes (\alpha';\beta')\ .$$

Given two Petri Categories $\underline{C}$ and $\underline{D}$ a Petri category morphism from $\underline{C}$ to $\underline{D}$ is a functor that is a monoid homomorphism when restricted to both the objects and the morphisms. This data determines a category $\underline{\mathsf{CatPetri}}$.

□

The equations are equivalent to saying that the operation $\otimes$ is a bifunctor $\otimes : \underline{C} \times \underline{C} \to \underline{C}$. Furthermore this implies that a Petri category is a monoidal category.

A Petri net N can be completed to a Petri category $\mathsf{T}(N)$ by the following proof rules:

$$\frac{u \text{ in } P^{\otimes}}{t_u : u \to u \text{ in } \mathsf{T}[N]} \tag{A.1}$$

$$\frac{\iota(t) = u, o(t) = v \text{ and } t\text{in } N}{t : u \to v \text{ in } \mathsf{T}[N]} \tag{A.2}$$

$$\frac{t_1 : u \to v, t_2 : v \to w \text{ in } \mathsf{T}[N]}{t_1; t_2 : u \to w \text{ in } \mathsf{T}[N]} \tag{A.3}$$

$$\frac{t : u \to v, t' : u' \to v' \text{ in } \mathsf{T}[N]}{t \otimes t' : u \otimes u' \to v \otimes v' \text{ in } \mathsf{T}[N]} \tag{A.4}$$

The rules define a functor $\mathsf{T}[\_\_] : \underline{\mathsf{Petri}} \to \underline{\mathsf{CatPetri}}$ which is left adjoint to the forgetful functor $\mathsf{U} : \underline{\mathsf{CatPetri}} \to \underline{\mathsf{Petri}}$.

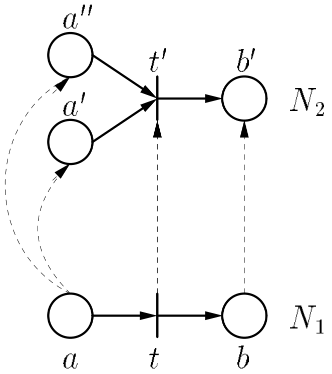
The category $\underline{\mathsf{CatPetri}}$ has several uses. First of all due to the existence of the left adjoint T we can think of the $\underline{\mathsf{CatPetri}}$ net $\mathsf{T}(N)$ as a kind of generalized behavior of the net. The $\underline{\mathsf{CatPetri}}$ net $\mathsf{T}(N)$ contains all possible behaviors of the net N. This kind of "abstract" behavior is studied in (Degano et al. 1989), where it is related to the different notions of process for Place/Transition systems (cf. (Best and Devillers 1987)). Secondly the notion of morphism is a very strong one and we shall have reason to examine it further in the sequel.

Actually the left adjoint T is the composition of several left adjoints, because there exists a sequence of left adjoints between the categories defined above:

$$\underline{\mathsf{Petri}} \to \underline{\mathsf{Petri}_0} \to \underline{\mathsf{CMonPetri}} \to \underline{\mathsf{CatPetri}}\,.$$

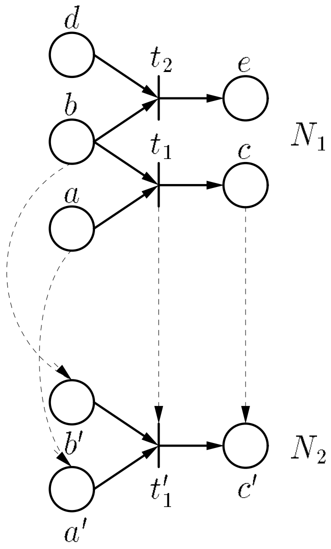
These left adjoints allow us to relate the different categories and their morphisms to each other.

A comparison of the different categories allows us to characterize the morphism by their possible actions on transitions.



$$\begin{aligned} f_p(a) &= a' \otimes a'' \\ f_p(b) &= b' \\ f_t(t) &= t' \end{aligned}$$

Figure A.1: A morphism in Petri.



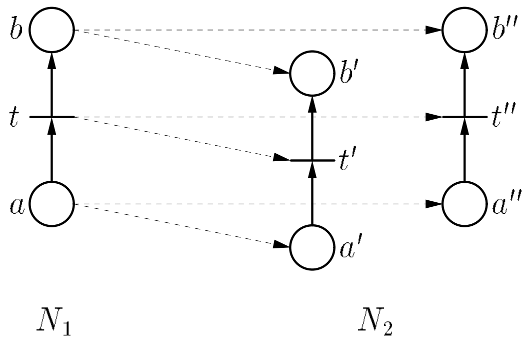
$$\begin{aligned} f_p(a) &= a' \\ f_p(b) &= b' \\ f_p(c) &= c' \\ f_p(d) &= 0 \\ f_p(e) &= 0 \\ f_t(t_1) &= t'_1 \\ f_t(t_2) &= 0 \end{aligned}$$

Figure A.2: A morphism in Petri_0.

- Petri: A transition is mapped to another transition (see figure A.1).
- Petri_0: We can erase parallel transitions (see figure A.2).
- CMonPetri: A transition is mapped onto parallel compositions of transitions (see figure A.3).
- CatPetri: A transition can be mapped onto an entire computation with sequential and parallel compositions of transitions (see figure A.4).

The morphisms are listed in their order of complexity. The sequence of left adjoints mentioned previously makes this order an inclusion in the sense that with a more complex morphism we can achieve all that can be achieved with the more simpler one.

All the morphisms obey the following important fact.



$$
\begin{aligned}
f_p(a) &= a' \otimes a'' \\
f_p(b) &= b' \otimes b'' \\
f_t(t) &= t' \otimes t''
\end{aligned}
$$

Figure A.3: A morphism in CMonPetri.

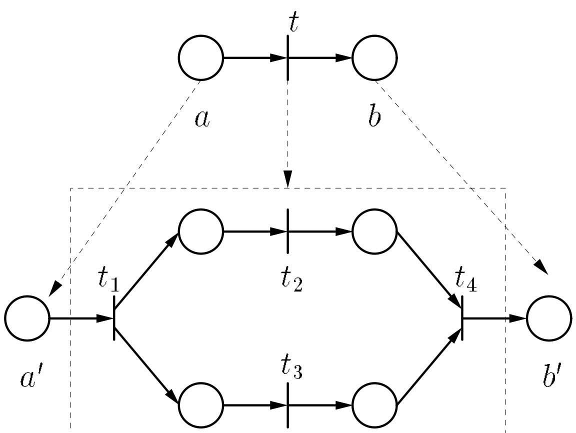


Figure A.4: A morphism in CatPetri.

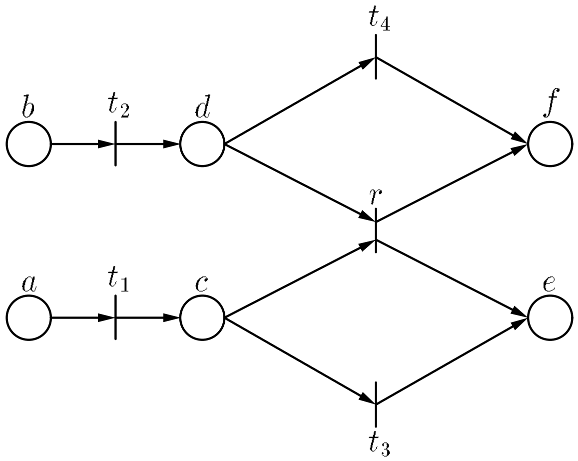


Figure A.5: The net N_1.

Fact A.2.6
The morphisms preserve behavior. That is, given markings M_1 and M_2 in N_1 s.th. M_2 is obtained from M_1 by firing t_1, there exists a transition $t_1' \in N_2$ s.th. $f(M_2)$ is obtained from $f(M_1)$ by firing t_1' and $t_1' = f(t_1)$. $\square$

This fact is a consequence of the fact that in <u>CatPetri</u> morphisms are functors and thus preserve parallel and sequential compositions of transitions, in other words the behavior.

Before ending this subsection we shall briefly discuss the morphisms in <u>CatPetri</u> so that the reader will get a feel for what we mean by behavior in $\mathsf{T}(N)$. We shall illustrate this by discussing refinement, as some morphisms in <u>CatPetri</u> can be interpreted as refinement morphisms. In the net N_1 in figure A.5 we wish to refine the transition r by the net R in figure A.6. A simple graphical substitution gives as a result the net N_2 in figure A.7, this is called an atomic refinement. Now there exists a morphism $f : \mathsf{T}(N_1) \to \mathsf{T}(N_2)$ in <u>CatPetri</u> given by the following assignments:

$$\begin{array}{rcl} t_1 & \mapsto & {t_1}' \\ t_2 & \mapsto & {t_2}' \\ t_3 & \mapsto & {t_3}' \\ t_4 & \mapsto & {t_4}' \\ r & \mapsto & ({t_6}' \otimes {t_7}'); {t_8}' \ . \end{array}$$

The net N_2 can thus be interpreted as a refinement of N_1. Unfortunately the morphism f does not preserve deadlock freeness, because both R and N_1 are deadlock free, but N_2 is not. This is because from the marking $c \otimes d$ both steps ${t_4}' \otimes {t_6}' : c' \otimes d' \to f' \otimes g'$ and ${t_3}' \otimes {t_7}' : c' \otimes d' \to e' \otimes h'$ and are possible. The

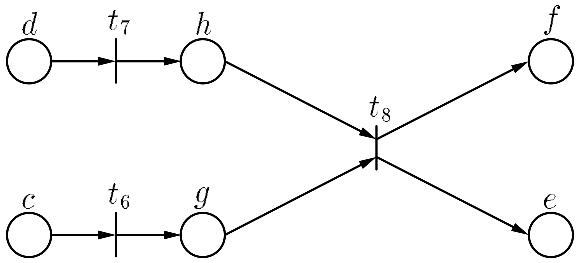


Figure A.6: The net R.

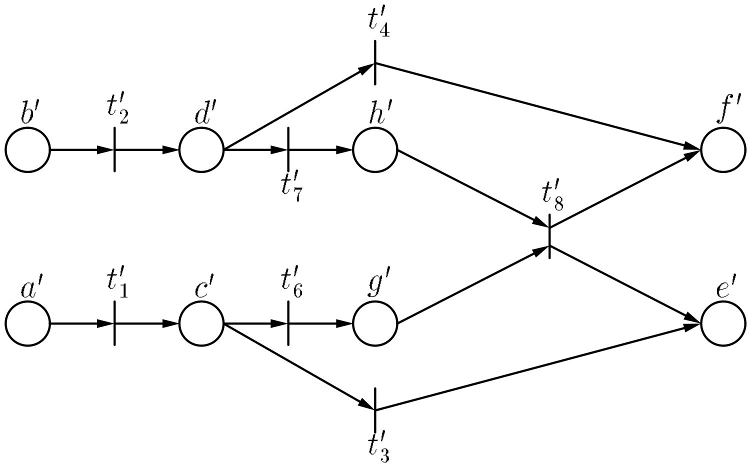


Figure A.7: The net N_2.

resulting markings are dead. The problem arises, because the net R exhibits a phenomenon called *initial concurrency*, where the initial transitions t_6' and t_7' need not fire concurrently. There are standard solutions to this problem (see e.g. (van Glabbeek 1990) p.191) where one requires that the refinement net shall obey conditions ensuring that it will act like a transition with respect to its environment:

- it cannot move without being activated by the environment,
- it has the same possible behaviors whenever it is activated,
- it may not deadlock,
- it consumes and produces tokens in a coincident manner.

The first condition means, that the net shall not be marked initially. The second condition means, that the net is not allowed to store tokens. The final condition prevents initial and final concurrency.

But there is a way in which to interpret the morphism f, so that it preservers deadlock-freeness. Because the morphism is in $\underline{\mathsf{CatPetri}}$ it is actually mapping behaviors to behaviors. Thus it can be interpreted as specifying that for the net N_2 to be a deadlock-free refinement of N_1 the transitions t_6', t_7', t_8 have to be fired in the order $(t_6' \otimes t_7'); t_8'$. Gorrieri and Montanari (1995) discuss the implementation of CCS (Milner 1989) through such morphisms.

A.2.2 Completeness properties.

The main aim of this subsection is to show which co-limits do exist in Petri. The fact that Petri does not have all co-limits will lead to definition of a category PetriG which is a subcategory of Petri that has all co-limits, and is used in chapters 2, 4 and 5

The lack of arbitrary limits and co-limits in the category Petri follows from the lack of the corresponding limits and co-limits in the category of free commutative monoids. This is illustrated by the lack of equalizers and coequalizers. The following two counter-examples are due to Josè Meseguer (Meseguer 1991). First equalizers:

Counterexample A.2.7
Take the following two maps from N^3 to N:

$$\begin{aligned} f(x,y,z) &= 7x + 2y + 5z \\ g(x,y,z) &= 2x + 5y + 7z \ . \end{aligned}$$

The equalizer of the pair f, g is the set:

$$E = \{(x,y,z) | 5x = 3y + 2z\},$$

which is generated by the set of vectors:

$$\begin{aligned} u_1 &= (1,1,1) \\ u_2 &= (3,5,0) \\ u_3 &= (3,1,6) \\ u_4 &= (4,0,10) \ . \end{aligned}$$

This set of vectors E is not a free monoid, because we have the identity

$$(6,2,12) = 2u_3 = 3u_1 + u_4 \ .$$

□

For coequalizers there is an analogous counterexample:

Counterexample A.2.8
Take the following two maps from N to N:

$$\begin{aligned} h_1(x) &= 3x \\ h_2(x) &= 0 \ . \end{aligned}$$

The coequalizer of these two maps is the following submonoid of N:

$$Q = \{n | n \ \text{ not divisible by } \ 3\},$$

which is clearly not free. □

This leaves us with the following two propositions.

Proposition A.2.9
The category Petri has all products.

Proof:
The net $\langle \{*\}, \phi, 0, 0 \rangle$ is the terminal object. The product $N_1 \times N_2$ (see figure A.8 of two nets N_1 and N_2 is

$$N_1 \times N_2 = \langle T_1 \times T_2, P_1 \uplus P_2, \iota, o \rangle,$$

where the projections are pairs $\pi^i = \langle \pi_T^i, \pi_P^i \rangle$ with $\pi_T^i : T_1 \times T_2 \to T_i$ and $\pi_P^i : P_1 \uplus P_2 \to P_i$ for $i = 1, 2$. The maps ι and o are defined by

$$\begin{aligned} \iota : \langle t_1, t_2 \rangle &\mapsto \iota_1(t_1) \otimes \iota_2(t_2) \\ o : \langle t_1, t_2 \rangle &\mapsto o_1(t_1) \otimes o_2(t_2) \ . \end{aligned}$$

□

Proposition A.2.10
The category Petri has all coproducts.

Proof:
The initial object is the empty net $\langle \phi, \phi, 0, 0 \rangle$. The coproduct $N_1 + N_2$ is the disjoint union of the two nets $N_1 + N_2 = \langle T_1 \uplus T_2, P_1 \uplus P_2, \iota_1 \uplus \iota_2, o_1 \uplus o_2 \rangle$. □

The idea that the empty net is the initial object may at first seem slightly puzzling, because an initial object in the category of monoids is any one object monoid where the object is the unit element. But in nets the unit element is to be thought of as the empty marking which means the empty set of places.

The fact that Petri lacks more useful co-limits makes it at first sight not very useful in its applications to compositionality. Fortunately there is a solution to this problem proposed by Hummert in (Hummert 1989). Recall that we

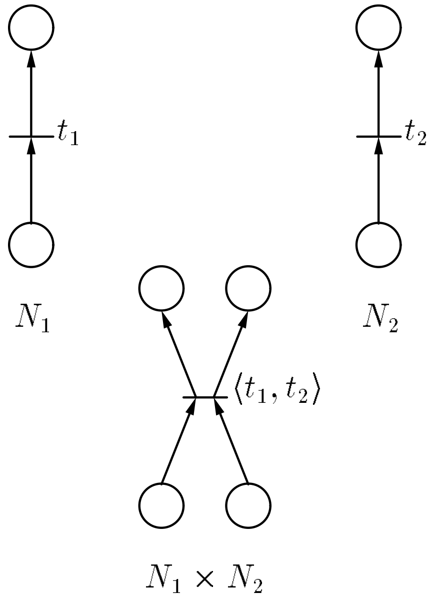


Figure A.8: $N_1 \times N_2$ is the product of N_1 and N_2.

started with the category of graphs to which we then added structure. The choice of morphism that was made is the reason that co-completeness was lost. By changing the morphism co-completeness can be regained.

Definition A.2.11
The category <u>PetriG</u> has as objects Petri nets and as morphism pairs of functions $\langle f_t, f_p \rangle$ such that

$$\begin{tikzcd}
T \arrow[r, shift left, "\iota"] \arrow[r, shift right, "o"'] \arrow[d, "f_t"'] & P^{\otimes} \arrow[d, "f_p^{\otimes}"] \\
T' \arrow[r, shift left, "\iota'"] \arrow[r, shift right, "o'"'] & P^{\otimes'}
\end{tikzcd}$$

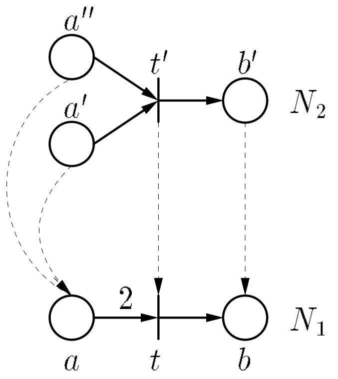
commutes. ($f_p^{\otimes}$ is the free extension of f_p to a monoid homomorphism.)

□

The following is obvious:

Proposition A.2.12
<u>PetriG</u> is a wide subcategory of <u>Petri</u>. □

An example will illustrate the difference between the morphisms. Figure A.9



$$
\begin{array}{lcl}
f_p(a') & = & a \\
f_p(a'') & = & a \\
f_p(b') & = & b \\
f_t(t') & = & t
\end{array}
$$

Figure A.9: A morphism in PetriG.

shows a map that is a map in PetriG. Its inverse is a map in Petri but not in PetriG.

Morphism in PetriG are essentially graph morphism and the category of graphs has all co-limits. Because the category PetriG is the subcategory of Petri that corresponds to the image of the free functor F(Graph) that is left adjoint to the forgetful functor U : Petri → Graph the category PetriG is co-complete. More explicitly the constructions are show below.

Theorem A.2.13
PetriG is co-complete

Proof:
We list the simple co-limits:

- the initial object is the net $\langle \phi, \phi, 0, 0, \rangle$,
- the coproduct is the juxtaposition of the two nets, with the obvious injections, and
- the coequalizer of $N_1 \rightrightarrows^{f}_{g} N_2$ is the net $N = \langle T, P, \iota, o \rangle$ with

$$
\begin{array}{lcl}
T & = & coeq(T_1 \rightrightarrows^{f_T}_{g_T} T_2) \text{ in } \underline{\text{Set}} \\
P & = & coeq(P_1 \rightrightarrows^{f_P}_{g_P} P_2) \text{ in } \underline{\text{Set}} .
\end{array}
$$

The maps ι, o are defined by the universal property of the coequalizer.

□

Unfortunately we do not have all products in PetriG. Consider the nets N_1, N_2, N_3 in figure A.10. If N_3 were to be the product of N_1 and N_2, this would imply the existence of projections $\pi^i = \langle \pi^i_T, \pi^i_P \rangle$ for $i = 1, 2$. But in PetriG there exist no maps $N_3 \rightarrow N_i$, because we need to preserve the arc-weights and this can't map $p_1 \otimes p_2$ onto p_i

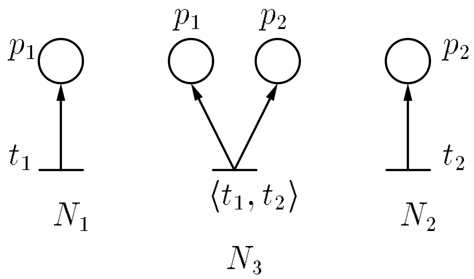


Figure A.10: The category <u>PetriG</u> does not have all products.

A.3 Universal algebra

A.3.1 Many-sorted algebra

The theory of abstract data-types is based on many-sorted algebra. A category-theoretic formulation is presented by Goguen and Burstall in (Goguen and Burstall 1984). In the following we shall follow the more concise formulation of (Goguen and Burstall 1990).

The fundamental intuition is, that a data-type consists of a set of elements of different sorts, and a set of operations making the data-type into an algebra. Each algebra has a signature that names the sorts and the operators of the algebra.

Definition A.3.1
Given a set of *sorts* S an *equational signature* is a pair $\langle S, \Sigma \rangle$, where Σ is a family of sets (of *operator* names) indexed by $S^* \times S$, where S^* denotes the set of strings with alphabet S. The operator $\sigma \in \Sigma_{w,s}$ has *arity* w, *sort* s, and *rank* or *type* $\langle w, s \rangle$. It is customary to just write Σ instead of $\langle S, \Sigma \rangle$. □

Definition A.3.2
A *equational signature morphism* $\Phi : \langle S, \Sigma \rangle \to \langle S' \Sigma' \rangle$ is pair $\langle h_S, h_\Sigma \rangle$ where $h_S : S \to S'$ is the sort map and h_Σ is a $S^* \times S$-indexed family of maps $h_{u,s} : \Sigma_{u,s} \to \Sigma'_{f^*(u),f(s)}$. □

Definition A.3.3
Equational signatures and equational signature morphisms form the *category of equational signatures* <u>Sig</u>. □

Definition A.3.4
Given a signature Σ, a Σ-algebra A is a S-indexed family of sets $|A| = \langle A_s | s \in S \rangle$ called the carries of A together with an $S^* \times S$ indexed family α of maps $\alpha_{u,s} : \Sigma_{u,s} \to [A_u \to A_s]$ for u in S^* and s in S, where $A_{s_1 \ldots s_n} = A_{s_1} \times \cdots \times A_{s_n}$ and $[A \to B]$ denotes the set of functions from A to B. □

Definition A.3.5
A Σ-homomorphism is a S-indexed map $h : A \to A'$ that satisfies the equation

$$h(\sigma(x_1, \ldots, x_n) = h(\sigma)(h(x_1), \ldots, h(x_n)) .$$

$\square$

Σ-algebras and Σ-homomorphism form a category $\underline{\mathsf{Alg}}_\Sigma$. There exists a *forgetful* functor $U : \underline{\mathsf{Alg}}_\Sigma \to \underline{\mathsf{Set}}_S$ which sends a Σ-algebra to its S-indexed set of carriers.

Given a S-indexed set of variables X we can form the set of all terms that are constructed with operator symbols from Σ and variables from X. This set forms a Σ-algebra in a natural way. It is called the *free* algebra over the set of variables X and denoted by $T_\Sigma(X)$. We first define a special case, defining $(T_\Sigma)_s$ to be the smallest set of strings of symbols such that:

- $\Sigma_{\lambda,s} \subseteq T_{\Sigma,s}$,
- $\sigma \in \Sigma_{s_1 \ldots s_n, s}$ and $t_i \in T_{\Sigma, s_i}$ imply that the string $\sigma(t_1, \ldots, t_n)$ is in $T_{\Sigma,s}$

If we now define α by

- for $\sigma \in \Sigma_{\lambda,s}$ let $\alpha(\sigma)$ be the string σ of length 1 in $T_{\Sigma,s}$,
- for $\sigma \in \Sigma_{s_1 \ldots s_n, s}$ and $t_i \in T_{\Sigma,s_i}$ let $\alpha(\sigma(t_1, \ldots, t_n))$ be the string $\sigma(t_1, \ldots, t_n)$ in $T_{\Sigma,s}$.

then α defines the Σ-structure on T_Σ. Let now define $\Sigma(X)$ to be the signature with $(\Sigma(X))_{\lambda,s} = (\Sigma)_{\lambda,s} \cup X_s$ and $(\Sigma(X))_{u,s} = (\Sigma)_{u,s}$ if $u \neq \lambda$. Then the free algebra $T_\Sigma(X)$ is just $T_{\Sigma(X)}$ seen as a Σ-algebra.

The free algebra enjoys the following universal property. For any Σ-algebra, every map $f : X \to U(B)$ has a unique extension to a Σ-homomorphism $f^\# : T_\Sigma(X) \to B$. The map $f : X \to U(B)$ is called *variable assignment* and its extension $f^\#$ *assignment.*

The construction of a free Σ-algebra on a set of variables X extends to a functor $F : \underline{\mathsf{Set}} \to \underline{\mathsf{Alg}}_\Sigma$ that is left adjoint to the forgetful functor $U : \underline{\mathsf{Alg}}_\Sigma \to \underline{\mathsf{Set}}$.

An abstract data-type is usually specified using a signature and a set of equations that specify the behavior of the operations.

Definition A.3.6
A Σ-equation e is a triple $\langle X, t_1, t_2 \rangle$, where X is a set of S-sorted variables, and $t_1, t_2 \in T_{\Sigma,s}(X)$ are terms of the same sort. Such an equation is usually written $t_1 = t_2$. $\square$

The idea that an equation "holds" in an algebra is formalized as follows.

Definition A.3.7
A Σ-algebra A satisfies a Σ-equation $t_1 = t_2$, written $A \models t_1 = t_2$ iff $\mathrm{ass}_A^{\#}(t_1) = \mathrm{ass}_A^{\#}(t_2)$ for every assignment $\mathrm{ass}_A^{\#} : X \to U(A)$. $\square$

An abstract data-type is now given by a presentation.

Definition A.3.8
A tuple $\langle S, \Sigma, EQ\rangle$, where $\langle S, \Sigma\rangle$ is a signature and EQ is a set of equations, is called a Σ-presentation. A Σ-algebra is a Σ-presentation algebra if $A \models e$ for all $e \in EQ$. $\square$

We will use the term Σ-algebra for short, both when Σ is a presentation and when it is a signature, if the exact meaning is clear from the context.

The equations of the presentation induce an equivalence relation on the terms of T_Σ. The resulting algebra is called a term algebra and denoted by $T_{\Sigma/\equiv}$.

Definition A.3.9
A presentation morphism $h_\Sigma : \langle S, \Sigma, EQ\rangle \to \langle S', \Sigma', EQ'\rangle$ is a signature morphism h_Σ, that is extended on equations by:

$$h_\Sigma^{\#}(\langle X, t_1, t_2\rangle) = \langle h_\Sigma^{\#}(X), h_\Sigma^{\#}(t_1), h_\Sigma^{\#}(t_2)\rangle \ .$$

$\square$

The category $\underline{\mathsf{Alg}}_\Sigma$ extends to a functor on the category $\underline{\mathsf{Sig}}$ that maps each signature Σ to the category $\underline{\mathsf{Alg}}(\Sigma)$ of all Σ-algebras.

Definition A.3.10
The functor $\underline{\mathsf{Alg}} \to \underline{\mathsf{Cat}}^{op}$ sends each signature Σ to the category $\underline{\mathsf{Alg}}_\Sigma$, and sends each signature morphism $\phi : \langle h_S : S \to S', h_\Sigma : \Sigma \to \Sigma'\rangle$ to the functor $\underline{\mathsf{Alg}}(\phi) : \underline{\mathsf{Alg}}_\Sigma \to \underline{\mathsf{Alg}}_{\Sigma'}$ that:

1. sends a Σ-algebra $\langle A', \alpha'\rangle$ to the Σ-algebra $\langle A, \alpha\rangle$ with $A_s = A'_{h_S(s)}$ and $\alpha = h_\Sigma ; \alpha'$, and
2. sends each Σ'-homomorphism $h' : A' \to B'$ to the Σ-homomorphism $\underline{\mathsf{Alg}}(\phi)(h') = h : \underline{\mathsf{Alg}}(\phi)(A') \to \underline{\mathsf{Alg}}(\phi)(B')$ defined by $h_s = h'_{h_S(s)}$.

$\square$

Definition A.3.11
Since the functor $\underline{\mathsf{Alg}}$ maps a Σ-algebra A' to a Σ-algebra A, given a signature morphism $h_\Sigma : \Sigma \to \Sigma$ there is a natural Σ-homomorphism $h_A : A \to A'|_{h_\Sigma}$ where $A'|_{h_\Sigma} = \underline{\mathsf{Alg}}(h_\Sigma)(A')$ that is called a "generalized homomorphism".

$\square$

A.3.2 Order-sorted Algebra

The idea of sub-sorts can be modeled by adding an ordering on the sorts of a many-sorted signature. This generalization leads to the definition of Order-sorted Algebra. The basic reference is (Goguen and Meseguer 1992).

Definition A.3.12
An *order-sorted signature* is a triple $\langle S, \leq, \Sigma \rangle$, where $\langle S, \Sigma \rangle$ is an equational signature, and $\langle S, \leq \rangle$ is a poset, and the operations satisfy the following monotonicity condition:

$$\sigma \in \Sigma_{w1,s1} \cap \Sigma_{w2,s2} \text{ and } w1 \leq w2 \text{ imply } s1 \leq s2 \; .$$

□

Definition A.3.13
Let $\langle S, \leq, \Sigma \rangle$ be an order-sorted signature. Then an $\langle S, \leq, \Sigma \rangle$-*algebra* is an $\langle S, \Sigma \rangle$-algebra A such that:

1. $s \leq s' \in S$ implies $A_s \subseteq A'_s$, and
2. $\sigma \in \Sigma_{w1,s1} \cap \Sigma_{w2,s2}$ and $w1 \leq w2$ imply $A_\sigma : A_{w1} \to A_{s1}$ equals $A_\sigma : A_{w2} \to A_{s2}$ on $w1$.

□

Definition A.3.14
Let $\langle S, \leq, \Sigma \rangle$ be an order-sorted signature, and let A, B be order-sorted $\langle S, \leq, \Sigma \rangle$-algebras. An $\langle S, \leq, \Sigma \rangle$-*homomorphism* is an $\langle S, \Sigma \rangle$-homomorphism such that

$$s \leq s' \text{ and } a \in A_s \text{ imply } h_s(a) = h'_s(a) \; .$$

$\langle S, \leq, \Sigma \rangle$-algebras and $\langle S, \leq, \Sigma \rangle$-homomorphisms form a category $\Sigma - \underline{\mathsf{OSAlg}}$.
□

To define the term algebra, the signature needs to satisfy the following regularity conditions that ensures the existence of a least rank for each term.

Definition A.3.15
An order-sorted signature Σ is *regular* iff given $\sigma \in \Sigma_{w1,s1}$ and given $w0 \leq w1$ in S^*, there is a least rank $\langle w, s \rangle \in S^* \times S$, such that $w0 \leq w$ and $\sigma \in \Sigma_{w,s}$.
□

The free algebra can now be defined by the following construction

- $\Sigma_{\lambda,s} \subseteq T_{\Sigma,s}$,

- $T_{\Sigma,s'} \subseteq T_{\Sigma,s}$ iff $s' \leq s$,
- $\sigma \in \Sigma_{s_1 \dots s_n,s}$ and $t_i \in T_{\Sigma,s_i}$ imply that the string $\sigma(t_1,\dots,t_n)$ is in $T_{\Sigma,s}$

Now since Σ is regular, by theorem 2.12 of (Goguen and Meseguer 1992), T_Σ is initial.

If one requires the sort order S to be *locally-filtered*, ie. for each connected component of the order, for any two elements $s, s' \in S$ there exists an element s'', such that $s, s' \leq s''$, one transform each order-sorted algebra A with signature Σ into a many sorted algebra $A^\#$ with signature $\Sigma^\#$, together with a set of conditional equation J. The idea is that we view each operator in Σ as an operator of $\Sigma^\#$, and whenever $s \leq s'$ we add an operator $c_{s,s'} \in \Sigma^\#_{s,s'}$ that is called *inclusion operator*. The conditional equations are the following:

1. (identity) $c_{s,s}(x) = x$ for each $s \in S$,
2. (injectivity) $x = y$ if $c_{s,s'}(x) = c_{s,s'}(y)$, for each $s \leq s' \in S$,
3. (transitivity) $c_{s',s''}(c_{s,s'}(x)) = c_{s,s''}(x)$, for each $s \leq s' \leq s'' \in S$,
4. (homomorphism) whenever $\sigma : s1 \dots sn \to s$ and $\sigma : s'1 \dots s'n \to s'$ are in Σ, with $si \leq s'i$, and $s \leq s'$ in S, then

$$c_{s,s'}(\sigma_{s1 \dots sn,s}(x_1,\dots,x_n)) = \sigma_{s'1 \dots s'n,s'}(c_{s1,s'1}(x_1),\dots c_{sn,s'n}(x_n)) \; .$$

The construction extends to a functor, and the result can now be stated as follows.

Theorem A.3.16 (Theorem 4.2 (Goguen and Meseguer 1992))
Given a coherent order-sorted signature Σ, the functor $(\_)^\# : \Sigma - \underline{\mathsf{OSAlg}} \to (\Sigma^\#, J) - \underline{\mathsf{Alg}}$ is an equivalence of categories. □

A.4 Monoids

Given a set S the free commutative monoid generated by S is denoted by $S^\otimes$. If $f : S \to S'$ is a $\underline{\mathsf{Set}}$-function, then the free extension along the functor $F : \underline{\mathsf{Set}} \to \underline{\mathsf{Mon}}$ is denoted by $f^\otimes$.

Definition A.4.1
Given a monoid $A^\otimes$ with a set of generators A, define a function $\gamma_A : A^\otimes \to \mathcal{P}(A)$ by the assignment $\gamma_A(a_1 \otimes \cdots \otimes a_n) = \{a_1,\dots,a_n\}$. γ maps a term of the monoid into its set of generators. If the context is clear the subscript will be omitted. □

Bibliography

Abadi, M. and Plotkin, G. D.: 1991, A logical view of composition and refinement (excerpt), *in* E. Best (ed.), *Third Workshop on Concurrency and Compositionality, Goslar*, pp. 1–10.

Abramsky, S.: 1988, Linear process logic.

Abramsky, S. and Vickers, S.: 1990, Quantales, observational logic, and process semantics, *Research Report DOC 90/1*, Department of Computing, Imperial College, London, England.

Andreoli, J.-M. and Pareschi, R.: 1990, Linear objects: Logical processes with built-in inheritance, *7th International Conference on Logic Programming, Jerusalem.*

Asperti, A. and Longo, G.: 1991, *Categories, Types, and Structures*, MIT Press.

Asperti, A. and Martini, S.: 1989, Projections instead of variables, a category theoretic approach to logic programming, *Proc. of the sixth International Conference on Logic Programming.*

Asperti, A., Gorrieri, R. and Ferrari, G.: 1990, Implicative formulae in the state-as-proposition analogy, *Conference on Principles of Programming Languages*, pp. 59–71.

Barr, M. and Wells, C.: 1985, *Toposes, Triples and Theories*, Vol. 278 of *Grundlehren der mathematischen Wissenschaften*, Springer Verlag, Berlin.

Battiston, E., Cindio, F. D., Mauri, G. and Rapanotti, L.: 1991, Morphisms and minimal models for OBJSA nets, *Proc. 12th International Conference on Application and Theory of Petri Nets*, Gjern, pp. 455–475.

Benabou, J.: 1968, Structures algébriques dans les catégories, *Cahiers Topologie Géom. Différentielle Catégoriques* **10**, 1–126.

Bergstra, J. A. and Tucker, J. V.: 1987, Algebraic specifications of computable and semicomputable data types, *Theoretical Computer Science* **50**, 137–151.

Best, E. and Devillers, R.: 1987, Sequential and concurrent behaviour in Petri net theory, *Theoretical Computer Science* **55**, 87–136.

Brown, C.: 1989a, Petri nets as quantales, *Technical Report ECS-LFCS-89-96*, Laboratory for Foundations of Computer Science, University of Edinburgh, Scotland.

Brown, C.: 1989b, Relating Petri nets to formulae of linear logic, *Technical Report ECS-LFCS-89-87*, Laboratory for Foundations of Computer Science, University of Edinburgh, Scotland.

Brown, C.: 1991, *Linear Logic and Petri Nets: Categories, Algebra and Proof*, PhD thesis, Department of Computer Science, University of Edinburgh, Scotland.

Brown, C. and Gurr, D.: 1990, A categorical linear framework for Petri nets, *5th Symposium on Logic in Computer Science*, IEEE Computer Press, pp. 208–219.

Burstall, R. and Goguen, J. A.: 1977, Putting theories together to make specifications, *in* R. Eddy (ed.), *5th International Joint Conference on Artificial Intelligence*, Department of Computer Science, Carnegie-Mellon University, USA, pp. 1045–1068.

Burstall, R. and Goguen, J. A.: 1990, The semantics of Clear, a specification language, *in* D. Bjørner (ed.), *Proceedings of 1979 Copenhagen Winter School on Abstract Software Specification*, Vol. 86 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pp. 292–332.

Bütler, R., Esser, R. and Mattman, R.: 1991, *A distributed simulator for high order Petri nets*, Vol. 483 of *Lecture Notes in Computer Science*, Springer Verlag, pp. 47–63.

Cerrito, S.: 1990, A linear semantics for allowed logic programs, *5th IEEE Symp. on Logic in Computer Science*, IEEE Press.

Corradini, A.: 1990, An algebraic semantics for transition systems and logic programming, *Technical Report TD-9/90*, Università degli Studi di Pisa, Dipartimento di Informatica.

Corradini, A. and Asperti, A.: 1993, A categorical model for logic programs: Indexed monoidal categories, *in* J. de Bakker, W.-P. de Roever and R. G. (eds), *Semantics: Foundations and Applications*, Vol. 666 of *Lecture Notes in Computer Science*, Springer Verlag, Berlin, pp. 110–137.

Corradini, A. and Montanari, U.: 1992, An algebraic semantics for structured transition systems and its application to logic program, *Theoretical Computer Science* **103**, 51–106.

de Paiva, V. C. V.: 1989a, The dialectica categories, *AMS Conference on Categories in Computer Science 1987, Boulder*, Vol. 92 of *Contemporary Mathematics*, American Mathematical Society, Providence, Rhode Island, USA, pp. 47–62.

de Paiva, V. C. V.: 1989b, Dialectica-like model of linear logic, *Category Theory and Computer Science*, number 389 in *Lecture Notes in Computer Science*, Springer Verlag, Berlin, pp. 341–356.

Degano, P., Meseguer, J. and Montanari, U.: 1989, Axiomatizing net computations and processes, *4th Symposium on Logic in Computer Science*, IEEE Computer Press.

Dimitrovici, C. and Hummert, U.: 1989, Kategorielle konstruktionen für algebraische Petrinetze, *Technical Report 23*, Fachbereich Informatik, Technische Universität Berlin.

Dimitrovici, C., Hummert, U. and Pétrucci, L.: 1990, The properties of algebraic net schemes in some semantics, *L.R.I Rap. de Recherce 539*, Univeriste de Paris-Sud.

Dimitrovici, C., Hummert, U. and Pétrucci, L.: 1991, Semantics, composition, and properties of algebraic high-level nets, *in* G. Rozenberg (ed.), *Advances in Petri Nets 1991*, Vol. 524 of *Lecture Notes in Computer Science*, Springer Verlag, Berlin, pp. 93–117.

Ehrig, H. and Mahr, B.: 1985, *Fundamentals of Algebraic Specification I: Equations and Initial Semantics*, Vol. 6 of *EATCS Monographs in Theoretical Computer Science*, Springer-Verlag, Berlin.

Ehrig, H., Padberg, J. and Ribero, L.: 1992, Algebraic High Level Nets: Petri Nets revisited, *Proceedings of the ADT-COMPASS Workshop*, Caldes de Malavella Spain.

Engberg, U. and Winskel, G.: 1994, Linear logic on Petri nets, *Technical Report RS-94-3*, BRICS, Department of Computer Science, University of Aarhus, Denmark.

Ferrari, G. L.: 1990, Unifying models of concurrency, *Technical Report TD-4/90*, Dipartimento di Informatica, Universita di Pisa.

Findlow, G.: 1991, Can skeletons really be used to detect deadlocks of Nets, *Proceedings of the Fourth International Workshop on Petri Nets and Performance Models*, IEEE Computer Society Press, pp. 198–203.

Findlow, G.: 1992, Obtaining deadlock-preserving skeletons for Coloured Nets, *in* K. Jensen (ed.), *Application and Theory of Petri Nets*, Vol. 616 of *Lecture Notes in Computer Science*, Springer Verlag, Berlin, pp. 173–192.

Gallier, J. H.: 1987, *Logic for Computer Science*, John Wiley & Sons, New York.

Genrich, H. J.: 1986, Predicate/transition nets, *Petri Nets: Central Models and Their Properties*, Vol. 254 of *Lecture Notes in Computer Science*, Springer Verlag, pp. 207–247.

Genrich, H. J., Lautenbach, K. and Thiagarajan, P. S.: 1980, Elements of general net theory, *in* W. Brauer (ed.), *Net Theory and Applications*, Vol. 84 of *Lecture Notes in Computer Science*, Springer Verlag, Berlin, pp. 21–163.

Girard, J.-Y.: 1986, The system F of variable types, fifteen years later, *Theoretical Computer Science* **45**(2), 159–192.

Girard, J.-Y.: 1987a, Linear logic, *Theoretical Computer Science* **50**, 1–102.

Girard, J.-Y.: 1987b, Quantifiers in linear logic, *Proc. of the SILFS Conference, Cesena, Italy.*

Girard, J.-Y.: 1989a, Geometry of interaction I: Interpretation of system F, *Logic Colloquium '88*, North-Holland, Amsterdam, pp. 221–260.

Girard, J.-Y.: 1989b, Towards a geometry of interaction, *AMS Conference on Categories in Computer Science 1987*, Vol. 92 of *Contemporary Mathematics*, American Mathematical Society, Providence, Rhode Island, USA, pp. 69–108.

Girard, J.-Y.: 1990, Geometry of interaction II: Deadlock-free algorithms, *COLOG 1988*, Vol. 417 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pp. 76–93.

Girard, J.-Y. and Lafont, Y.: 1987, Linear logic and lazy computation, *TAPSOFT '87*, Springer LNCS 250, pp. 52–66.

Girard, J.-Y., Lafont, Y. and Taylor, P.: 1989, *Proofs and Types*, Vol. 7 of *Cambridge Tracts in Theoretical Computer Science*, Cambridge University Press, England.

Gödel, K.: 1958, Über eine bisher noch nicht benützte erweiterung des finiten standpunktes, *Dialectica* **12**, 280–287.

Goguen, J.: 1988, What is unification? A categorial view of substitution, equation and solution, *Research Report SRI-CSL-88-2R*, Computer Science Laboratory, SRI International.

Goguen, J. A.: 1973, Categorical foundations for general systems theory, *Advances in Cybernetics and Systems Research*, Transcripta Books, pp. 121–130.

Goguen, J. A.: 1975, Objects, *International Journal of General Systems* **1**, 237–243.

Goguen, J. A.: 1992, Sheaf semantics for concurrent interacting objects, *Mathematical Structures in Computer Science* **2**(2), 159–192.

Goguen, J. A. and Burstall, R.: 1984, Some fundamental algebraic tools for the semantics of computation. Part1: Comma categories, colimits, signatures and theories, *Theoretical Computer Science* **31**, 175–209.

Goguen, J. A. and Burstall, R.: 1990, Institutions: Abstract model theory for specification and programming, *Research Report ECS-LFCS-90-106*, Laboratory for Foundations of Computer Science, University of Edinburgh, Scotland.

Goguen, J. A. and Ginali, S.: 1978, A categorical approach to general systems theory, *in* G. Klir (ed.), *Applied General Systems Research*, Plenum, pp. 257–270.

Goguen, J. A. and Meseguer, J.: 1992, Order-sorted algebra I: equational deduction for multiple inheritance, overloading exceptions and partial operations, *Theoretical Computer Science* **105**, 217–273.

Gorrieri, R. and Montanari, U.: 1995, On the implementation of concurrent calculi in net calculi: Two case studies, *Theoretical Computer Science*.

Gray, J. W.: 1979, Fragments of the history of sheaf theory, *in* M. P. Fourman, C. J. Mulvey and D. S. Scott (eds), *Applications of Sheaves*, number 753 in *Lecture Notes in Mathematics*, Springer Verlag, Berlin, pp. 1–79.

Gray, J. W.: 1987, Categorical aspects of data type constructors, *Theoretical Computer Science* **50**, 103–135.

Grönberg, P., Tiusanen, M. and Varpaaniemi, K.: 1993, PROD - A Pr/T-Net reachability analysis tool, *Technical Report B11*, Digital Systems Laboratory, Helsinki University of Technology, Espoo.

Gunter, C. A. and Gehlot, V.: 1989, Nets as tensor theories, *in* G. D. Michelis (ed.), *10th International Conference on Applications and Theory of Petri Nets*, Bonn, pp. 174–191.

Harland, J. and Pym, D.: 1990, The uniform proof-theoretic foundation of linear logic programming, *Technical Report ECS-LFCS-90-124*, Laboratory for Foundations of Computer Science, University of Edinburgh, Scotland.

Hummert, U.: 1989, *Algebraische Theorie von High-level Netzen*, PhD thesis, Technische Universität Berlin.

Husberg, N.: 1992, *High Level Distributed Transition Systems in Categories*, PhD thesis, Helsinki University of Technology, Department of Computer Science.

Jensen, K.: 1986, Coloured Petri nets, *Petri Nets: Central Models and Their Properties*, Vol. 254 of *Lecture Notes in Computer Science*, Springer Verlag, pp. 247–299.

Lafont, Y.: 1988, The linear abstract machine, *Theoretical Computer Science* **59**, 157–180.

Lawvere, F. W.: 1963, *Functorial Semantics of Algebraic Theories*, PhD thesis, Columbia University.

Lilius, J.: 1991, On the compositionality and analysis of algebraic high-level nets, *Research Report A16*, Digital Systems Laboratory, Helsinki University of Technology, Espoo.

Lilius, J.: 1992, High-level nets and linear logic, *in* K. Jensen (ed.), *Application and Theory of Petri Nets*, Vol. 616 of *Lecture Notes in Computer Science*, Springer Verlag, Berlin, pp. 310–319.

Lilius, J.: 1993, A sheaf semantics for Petri nets, *Research report A23*, Helsinki University of Technology, Digital Systems Laboratory, Espoo.

Lilius, J.: 1994a, Folding place/transition nets, *in* P. Starke (ed.), *Proceedings of Workshop on Concurrency, Specification & Programming*, Humbolt Universität, Berlin.

Lilius, J.: 1994b, On the folding of algebraic nets, *Research Report A30*, Digital Systems Laboratory, Helsinki University of Technology, Espoo.

MacLane, S.: 1971, *Categories for the Working Mathematician*, Springer Verlag, Berlin.

Marti-Oliet, N. and Meseguer, J.: 1989, From Petri nets to linear logic, *in* D. P. et al. (ed.), *Category Theory and Computer Science*, number 389 in *Lecture Notes in Computer Science*, Springer Verlag, Berlin, pp. 313–340.

Masseron, M., Tollu, C. and Vauzeilles, J.: 1990, Generating plans in linear logic, *in* K. V. Nori and C. E. Veni Madhavan (eds), *Foundations of Software Technology and Theoretical Computer Science*, Vol. 472 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pp. 63–75.

Meseguer, J.: 1991, Private Communication.

Meseguer, J. and Montanari, U.: 1990, Petri nets are monoids, *Information and Computation* **88**, 105–155.

Milner, R.: 1989, *Communication and Concurrency*, Prentice Hall, London.

Milner, R., Tofte, M. and Harper, R.: 1990, *The definition of Standard ML*, The MIT Press.

Mulvey, C. J.: 1986, &, *Rendiconti del Circolo Matematico di Palermo* **12**, 99–104.

Murata, T. and Zhang, D.: 1988, A predicate-transition net model for parallel interpretation of logic programs., *IEEE Transactions on Software Engineering* **14**(4), 481–498.

Niefield, S. B. and Rosenthal, K. I.: 1988, Constructing locales from quantales, *Mathematical Proceedings of the Cambridge Philosophical Society* **104**, 215–234.

Padberg, J.: n.d., Personal communication.

Petri, C. A.: 1962, Kommunikation mit automaten, *Schriften des IIM nr. 3*, Bonn: Institut für Instrumentelle Mathematik.

Petri, C. A.: 1973, Concepts of Net Theory, *Mathematical Foundations of Computer Science*, Math. Inst. of the Slovak Acad. of Science, pp. 137–146.

Reisig, W.: 1986, Place/transition systems, *Petri Nets: Central Models and Their Properties*, Vol. 254 of *Lecture Notes in Computer Science*, Springer Verlag, Berlin, pp. 117–141.

Reisig, W.: 1991, Petri nets and algebraic specifications, *Theoretical Computer Science* **80**, 1–34.

Reisig, W. and Vautherin, J.: 1987, An algebraic approach to high level Petri nets, *8th Workshop on Applications and Theory of Petri Nets*, Zaragoza, Spain, pp. 51–72.

Smith, E. and Reisig, W.: 1987, The semantics of a net is a net: An exercise in general net theory, *in* K. Voss, H. J. Genrich and G. Rozenberg (eds), *Concurrency and Nets*, Springer Verlag, Berlin, pp. 461–497.

Thiagarajan, P. S.: 1986, Elementary net systems, *Petri Nets: Central Models and Their Properties*, Vol. 254 of *Lecture Notes in Computer Science*, Springer Verlag, Berlin, pp. 26–49.

van Glabbeek, R. J.: 1990, *Comparative Concurrency Semantics and Refinement of Actions*, PhD thesis, Vrije Universiteit te Amsterdam.

Vautherin, J.: 1987, Parallel systems specification with Coloured Petri Nets and algebraic specifications, *in* G. Rozenberg (ed.), *Advances in Petri Nets*, Vol. 266 of *Lecture Notes in Computer Science*, Springer Verlag, Berlin, pp. 293–308.

Wadler, P.: 1990, Linear types can change the world!, *in* M. Broy and C. B. Jones (eds), *Programming Concepts and Methods*, Elsevier Science Publisher B.V. (North-Holland), Amsterdam, pp. 561–580.

Winskel, G.: 1985, Petri nets, algebras and morphisms, *Technical Report 79*, Computer Laboratory, University of Cambridge, England.

Wolfram, D. A. and Goguen, J. A.: 1991, A sheaf semantics for FOOPS expressions, *Proceedings, ECOOP'91 Workshop on Object-based Concurrent Computation*.

Yetter, D. N.: 1990, Quantales and (noncommutative) linear logic, *Journal of Symbolic Logic* **55**(1), 41–64.