HELSINKI UNIVERSITY OF TECHNOLOGY
DIGITAL SYSTEMS LABORATORY
Series **A:** Research Reports ISSN 0783–5396
No. **16**; March 1991 ISBN 951–22–0828-8

ON THE COMPOSITIONALITY AND ANALYSIS OF ALGEBRAIC 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
SF–02150 ESPOO, FINLAND

HELSINKI UNIVERSITY OF TECHNOLOGY
DIGITAL SYSTEMS LABORATORY
Series **A:** Research Reports
No. **16**; March 1991

ISSN 0783–5396
ISBN 951–22–0828-8

On the Compositionality and Analysis of Algebraic High-Level Nets

Johan Lilius

Abstract: This work discusses three aspects of net theory: compositionality of nets, analysis of nets and high-level nets.

Net theory has often been criticised for the difficulty of giving a compositional semantics to a net. In this work we discuss this problem form a category theoretic point of view. In category theory compositionality is represented by colimits. We show how a high-level net can be mapped into a low-level net that represents its behaviour. This construction is functorial and preserves colimits, giving a compositional semantics for these high-level nets. Using this semantics we propose some proof rules for compositional reasoning with high-level nets.

Linear logic is a recently discovered logic that has many interesting properties. From a net theoretic point of view its interest lies in the fact that it is able to express resources in an analogous manner to nets. Several interpretations of Petri nets in terms of linear logic exist. In this work we study a model theoretic interpretation where the behaviour of the net gives a model of linear logic. This construction is extended to cover the algebraic high-level nets defined in this work.

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

Printing: TKK Monistamo; Otaniemi 1994

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

Phone: $\dfrac{90}{+358\text{–}0}$ 4511

Telex: 125 161 htkk sf

E-mail: lab@hutds.hut.fi

Contents

1 Introduction

In this work we discuss three ingredients of net theory: the problem of compositionality, the analysis of nets and high-level nets. These three ingredients are bound together by the use of category theory.

One of the main criticisms often raised against net theory is the fact that unlike for example for the process algebra CCS [53], there does not exists a formulation of a compositional semantics that has been generally accepted. By a compositional semantics we mean a semantics that comes equipped with some operations. These semantical operations must have an interpretation in terms of nets in the following way. When the operation is applied to subnets it gives rise to a compound net, and the semantics of this compound net can be calculated by applying the correspondig semantic operation on the semantics of the subnets. Examples of compositional operations in CCS are parallel composition and choice.

As a consequence net theorists have developed other methods for reasoning about properties of nets. Two methods that can be used are invariant analysis and reachability analysis. Invariant analysis is a good method to reason about properties that can be deduced from the static structure of the net. When one wants to talk about the specific behaviour of a net one has to resort to reachability analysis. Reachability analysis is a technique that involves a database of states and instances of transition firings and a corresponding query language. Recently a logic, called linear logic [28], that seems suitable as a query language has been discovered. One of our aims is to assess the usefulness of this logic for reasoning about high level nets.

In practice it has been noted that the modeling of systems is best done with high-level net formalisms. We wish to use category theory to apply ideas on compositionality and analysis to high-level nets, because category theory has been specifically developed as a tool to compare and bind together different areas in mathematics. In this work category theory will be used to talk about nets, algebraic specifications and logics.

The results presented in this work are as follows. We propose rules for compositional reasoning with high-level nets using colimits. We establish a functor from a category of high-level nets to a category of models for linear logic. The model in the image of the functor represents the behaviour of the net. Using this functor we propose an interpretation for the predicate version of linear logic.

As our approach to compositionality is based on category theory we need a notion of net morphism. There exists a classical definition of net morphism for Place/Transition systems [27], but it has one deficiency, it does not preserve the dynamics [65]. Why should a morphism preserve the dynamics the reader may now ask? On a philosophical level, morphisms between objects in mathematics are maps that preserve the structure of these objects. It seems plausible to choose the behaviour as the structure to preserve, also in view of the fact that this will allow us to relate two nets by their behaviour, and not simply by their graphical representation. Apart from the more philosophical arguments this choice of morphism has some other arguments that speak for it.

First of all it is possible to study the preservation of properties of nets by these morphisms. For example it is easy to deduce that the preservation of behaviour implies the preservation of quasiliveness and termination and the reflection of boundedness and deadlocks. For a proof of these properties consult e.g. [61].

Secondly, as category theoretic constructions are mainly concerned with morphisms, it is possible to study the preservation of properties by these constructs. Then because these constructions are rules for creating new objects from old ones it is possible to derive rules for reasoning about whole nets by reasoning about their parts [66].

When defining a category of nets one has to settle on a suitable mathematical representation for the multisets of places and possibly transitions and the flow relation in the net. Usually the flow relation in the net is represented by a pair of maps from the transitions to the places, a well known representation for graphs. A net morphism is then a pair of maps between the places and transitions of the nets respectively. Finally a commutativity condition is imposed on the maps. The approaches then mainly differ on the structure imposed on the sets of transitions and places and on the kind of maps between these. Also one or the other of the net morphism maps may be turned around.

In the approach taken by Winskel [65, 64, 66, 67] transitions and places are represented by vectors, and a net morphism consists of a pair of matrices that are interpreted as relations. Meseguer and Montanari [52] suggest monoids as the structure on the transitions and places. This means that a net morphism consists of a pair of monoid homomorphism. A recent idea by Carolyn Brown and Doug Gurr [11] is to turn the map on the places around and relax the commutativity requirement to a requirement of lax naturality [46]. The flow relation is represented by a pair of subobjects of the cartesian product of the set of places with the set of transtions. Finally Husberg [43] defines a multiset as a product. All these choices give rise to different kinds of constructions on nets.

The model proposed by Winskel has many commonalities with the process algebra CCS [53]. Indeed CCS processes were one of the main motivations in this work [68]. One way to see this is by observing how close the product in Winskels category is to the synchronisation construct in CCS. In CCS a synchronisation is not compulsory. This is reflected in the product through the fact that the original transitions are left in the net with the new synchronised transition.

On the other hand the approach taken by Meseguer and Montanari gives us a rich hierarchy of categories of nets. The categories differ in the amount of structure imposed on the transitions. The simplest category is the category of Petri nets, while the most complex is the category of Petri categories, where a Petri net is extended to a category. By working with groups instead of monoids the completeness properties of some of these categories have been studied by Dimitrovici et al. in [20, 19, 21, 18]. These results have then been applied by Hummert in his dissertation [42] to define a notion of net module.

The most novel approach is the one taken by Carolyn Brown and Doug Gurr. It is based on the Dialectica Categories of Valeria de Paiva [15, 16] which are a category theoretic model of Gödels "Dialectica interpretation" of higher order arithmetic [37]. The constructs available in these categories give rise to very interesting constructions on nets that have a connection with linear logic. Unfortunately the approach currently only works with elementary nets.

We have chosen to work with the categories of Meseguer and Montanari. The main reason for this is the existence of a rich hierarchy of categories that are related by left adjoints. This hierarchy allows us to choose the right level of abstraction at each point of the discussion. As we are interested in using colimits to specify compositional proof rules and colimits are preserved by left adjoints there is an added bonus in this approach.

The second ingredient of this work are high-level nets. These are nets, where tokens are allowed some individuality and the inscriptions on the net are taken from some formal system other than the natural numbers. Several different models of high-level nets exists. The most influential models are certainly the Predicate/transition nets by Genrich [26] and the Coloured Petri nets by Jensen [44]. When modelling systems in practice these high-level models have been found indispensable. Systems often consist of several processes that are run with different parameters. With high-level nets it is possible to give one net for all processes and model the different parameters with individual tokens. When modeling with low-level nets one would have to draw each different run by a net of its own. This is also known as the "football field" syndrome. For an example, the reader may consult pictures 11 and 12 on page 22, where figure 12 is the low-level equivalent of figure 11.

The use of algebraic specification as a specification method for sequential systems is well known [22]. The idea to combine algebraic specifications with

Petri nets has been put forward by several people (c.f. e.g. [61, 56, 57, 20, 19, 21, 18]). The motivation for this approach is that the specification of a distributed system consists of two parts: the specification of the data to be distributed into the system, and the paths that the data will take. The use of algebraic specification for the data and Petri nets for the dynamics then suggests itself. The approaches mainly differ in the amount of explicit category theory introduced.

The method used by Reisig in [57] and Reisig and Vautherin in [56] could be termed the initial algebra approach. The meaning (semantics) of a specification is there defined as any algebra which is isomorphic to the quotient term algebra of the specification. Dimitrovici et al. [61, 20, 19, 21, 18] allow the interpretation in any algebra of the specification. Some consequences of this are explored in [18]. Here the emphasis is not so much on the exact structure of the specifications as on the category theoretic properties of the net specifications.

Our definition of high-level net is modelled after [56]. We have added types to the algebraic specification as this seems the natural thing to do. For the semantics we have chosen the approach by Dimitrovici et al. The idea is that each high-level net can be mapped into a low level net that represents the behaviour of the high-level net.

The third and final ingredient is the analysis of nets. We are looking for a logic to describe properties of nets. Linear logic is a logic discovered by J-Y. Girard [28]. It can be thought of as a resource conscious logic and has thus recently gained considerable interest from the computer science community. Applications include implementation models for functional languages [34, 47, 63], the study of process algebras [3, 2, 1], logic programming [4, 14, 40], planning in AI [50], and net theory [5, 9, 10, 11, 8, 24, 39, 49]. The list is by no means meant to be comprehensive because the field is advancing very rapidly.

The applications relevant to our purposes are the applications to net theory. The aim in these studies is twofold. Linear logic is a logic with many new connectives that are difficult to understand intuitively from a logical point of view, as they arise in a purely formal manner. By giving an interpretation of linear logic in terms of Petri nets one tries to give an intuitive meaning to the connectives in terms of resources. But on the other hand if one tries to axiomatise Petri nets in classical logic one soon runs into technical difficulties due to the so called "frame problem". This problem does not arise when axiomatising Petri nets within linear logic. Thus the study of the connections between linear logic and Petri nets may benefit both areas of research.

There are different ways in which one can tackle the problem. The current approaches can be classified as proof theoretic [10, 39], model theoretic [9, 24] and category theoretic [11, 49]. In the proof theoretic approach the idea is to try to relate computations in nets to proofs of linear logic formulae. Brown [10] 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 [39] view a net as a theory. They are able to relate the cut-elimination of a proof (c.f. [25] sec. 6) to the notion of maximally concurrent process.

The model theoretic approaches of Brown [9] and Engberg and Winskel [24] are essentially the same. The idea is to generate a model of linear logic from the behaviours of a net. This is the approach that we will pursue in chapter 4.

The category theoretic approaches can really not be compared as their aims are completely different. The idea in [49] 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 [11] has already been discussed while discussing categories of nets. The interesting thing about their approach is the fact that the constructions they define can be interpreted as linear logic connectives. The approach could be summed up in the slogan "nets are linear logic propositions".

The structure of this work is as follows. First we discuss our choice of model for the low-level nets and prove some completeness and cocompleteness results for them. Then we present the notion of algebraic net. The semantics of an algebraic net is defined as a functor into a category of Place/Transition systems. Preservation of colimits by this semantics is studied. This property is then used to present some proof rules for combining algebraic nets. In the final part of this work we give an interpretation of linear monadic predicate logic in terms of the behaviour of the algebraic nets. We then close off with some discussion and an outlook to future research.

It is assumed that the reader has a general knowledge of net theory, category theory and the theory of algebraic specification. The appendix contains a short introduction to the rudiments of category theory. Other concepts are introducted as needed in the main body of the text.

2 Petri Nets are Monoids

In this section 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. The model has been proposed by Meseguer and Montanari in [52]. It is based on the observation that the markings of the net obey a commutative monoidal law. This monoidal structure is induced on the markings by the firing rule, in the sense that if the firing of a transition induces a change δ in the marking, the double firing (if possible !) of the same transition induces a change $2 * \delta$ in the marking. Thus we see that there is a natural monoidal structure on the markings. The observation to make now is, that a single transition with some input and output places can be viewed as a net. This net has two possible markings, the input places are marked or the output places are marked. Each of these markings correspond to an element of the free monoid on the set of places of the net. The input and output weighting functions can now be interpreted as monoid homomorphisms.

The usefulness of viewing the set 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 behaviour of the net is a much more abstract concept. The behaviour of the net includes *all* the possible behaviours of the net.

It has been proved by Winskel [65], that coproducts do not exists in the category of Petri nets with initial markings. All our categories now have coproducts, because they lack initial markings. But in these cases the coproduct merely consists of the juxtaposition of the individual nets with no connections between the components. So the lack of initial marking makes the coproduct something different than non-deterministic choice. There are two solutions to this problem. The first one is obviously to put the initial markings back into the net in some way. This can be done in a very elegant way by restricting the structure of the initial marking as has been proposed by Meseguer and Montanari [52]. The less obvious one is to look for some other colimits that would allow us to implement non-deterministic choices. As we shall see this is also possible. The investigation of other colimits is of interest also in view of the fact that the product and coproduct of nets with initial markings are rather arbitrary. They do not allow for a more controlled compositional construction of nets. We shall be interested in colimits that could allow us to control the extent of synchronisation of the nets in an elegant way.

This section 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 critical discussion of the use of the existing colimits will be done in section 3.4.

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. The importance of these adjunctions will become apparent in section 3.4 where we will make good use of the fact that left adjoints preserve colimits. Finally we shall try to give some characterising 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 [52].

Definition 1
The category of graphs $\underline{\text{Graph}}$ has as objects graphs $\langle T, P, \iota, o \rangle$ 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 $\underline{\text{Petri}}$.

Definition 2
The category of *Petri nets* $\underline{\mathrm{Petri}}$ 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.
$\square$

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 3
The category of *pointed Petri nets* $\underline{\mathrm{Petri}_0}$ 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. $\square$

If we add a monoidal structure to the transitions we get the category of Petri commutative monoids.

Definition 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 $\underline{\mathrm{CMonPetri}}$. $\square$

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 synchronisation 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 $\underline{\mathrm{CMonRPetri}}$, $\underline{\mathrm{RPetri}}$, $\underline{\mathrm{RGraph}}$.

The last category in the hierarchy is the category of Petri categories.

Definition 5
A *Petri category* is a small category $\underline{C} = (P^{\otimes}, T, ;, id)$ 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 $\underline{\mathrm{C}}$ and $\underline{\mathrm{D}}$ a Petri category morphism from $\underline{\mathrm{C}}$ to $\underline{\mathrm{D}}$ is a functor that is a monoid homomorphism when restricted to both the objects and the morphisms. This data determines a category $\underline{\mathrm{CatPetri}}$. □

The equations are equivalent to saying that the operation $\otimes$ is a bifunctor $\otimes : \underline{\mathrm{C}} \times \underline{\mathrm{C}} \to \underline{\mathrm{C}}$. Furthermore this implies that a Petri category is a monoidal category.

A Petri net can be completed to a Petri category by the following proof rules:

$$\frac{u \ \text{in} \ P^{\otimes}}{t : u \to u \ \text{in} \ \mathcal{T}[N]},$$

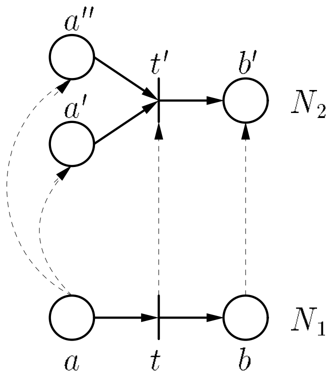
$$\frac{\iota(t) = u \ \text{and} \ o(t) = v \ \text{in} \ N}{t : u \to v \ \text{in} \ \mathcal{T}[N]},$$

$$\frac{t_1 : u \to v, t_2 : v \to w \ \text{in} \ \mathcal{T}[N]}{t_1 \circ t_2 : u \to w \ \text{in} \ \mathcal{T}[N]},$$

$$\frac{t : u \to v, t' : u' \to v' \ \text{in} \ \mathcal{T}[N]}{t \otimes t' : u \otimes u' \to v \otimes v' \ \text{in} \ \mathcal{T}[N]}\ .$$

The rules define a functor $\mathcal{T}[\_\_] : \underline{\mathrm{Petri}} \to \underline{\mathrm{CatPetri}}$ which is left adjoint to the forgetful functor $\mathcal{U} : \underline{\mathrm{CatPetri}} \to \underline{\mathrm{Petri}}$.

The category $\underline{\mathrm{CatPetri}}$ has several uses. First of all due to the existence of the left adjoint $\mathcal{T}$ we can think of the $\underline{\mathrm{CatPetri}}$ net $\mathcal{T}[N]$ as a kind of generalised behaviour of the net. The $\underline{\mathrm{CatPetri}}$ net $\mathcal{T}[N]$ contains all possible behaviours of the net N. This kind of "abstract" behaviour is studied in [17], where it is related to the different notions of process for Place/Transition systems (cf.



$$\begin{aligned} f_p(a) &= a' \otimes a'' \\ f_p(b) &= b' \\ f_t(t) &= t' \end{aligned}$$

Figure 1: A morphism in Petri.

[7]). 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 $\mathcal{T}$ is the composition of several left adjoints, because there exists a sequence of left adjoints between the categories defined above:

$$\underline{\text{Petri}} \rightarrow \underline{\text{Petri}_0} \rightarrow \underline{\text{CMonPetri}} \rightarrow \underline{\text{CatPetri}}\ .$$

These left adjoints allow us to relate the different categories and their morphisms to each other.

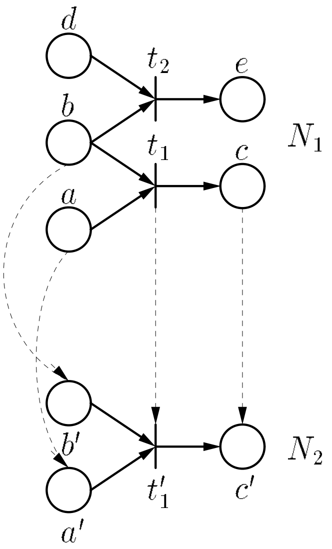
A comparison of the different categories allows us to characterise the morphism by their possible actions on transitions.

- Petri: A transition is mapped to another transition (see figure 1).
- Petri$_0$: We can erase parallel transitions (see figure 2).
- CMonPetri: A transition is mapped onto parallel compositions of transitions (see figure 3).
- CatPetri: A transition can be mapped onto an entire computation with sequential and parallel compositions of transitions (see figure 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.

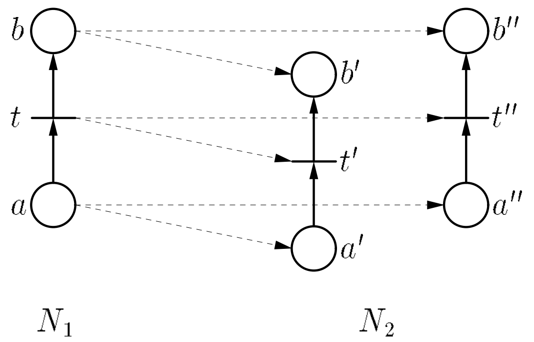
All the morphisms obey the following important theorem.

Theorem 1
The morphisms preserve behaviour. That is given a marking in N_1 there exists a corresponding marking in N_2. $\square$



$$
\begin{aligned}
f_p(a) &= a' \\
f_p(b) &= b' \\
f_p(c) &= c' \\
f_p(d) &= 0 \\
f_p(e) &= 0 \\
f_t(t_1) &= t_1' \\
f_t(t_2) &= 0
\end{aligned}
$$

Figure 2: A morphism in Petri$_0$.



$$
\begin{aligned}
f_p(a) &= a' \otimes a'' \\
f_p(b) &= b' \otimes b'' \\
f_t(t) &= t' \otimes t''
\end{aligned}
$$

Figure 3: A morphism in CMonPetri.

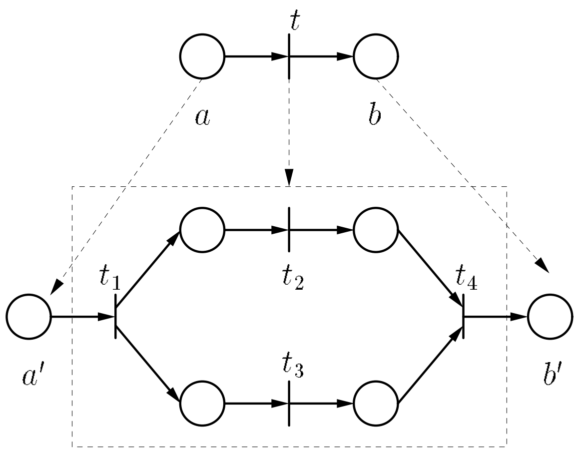


Figure 4: A morphism in CatPetri.

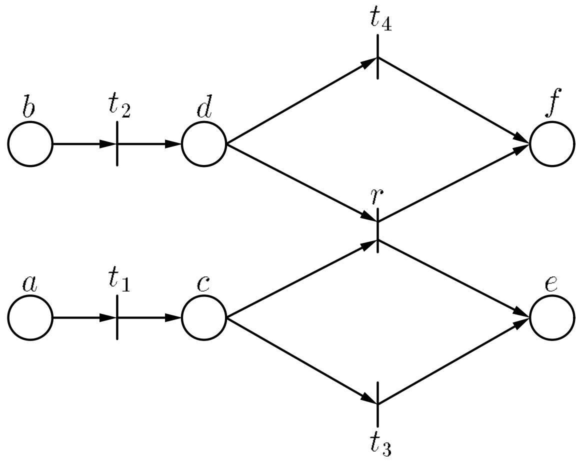


Figure 5: The net N_1.

This fact is a consequence of the fact that in CatPetri morphisms are functors and thus preserve parallel and sequential compositions of transitions, in other words the behaviour. Now, through the sequence of left adjoints, this result transfers to the other morphisms too.

To compare the morphisms discussed here with the morphisms defined by Winskel in [66] the following observations can be made:

- a Petri morphism corresponds to a synchronous morphism,
- a Petri_0 morphism corresponds to an asynchronous morphism, and
- a CMonPetri morphism corresponds to a homomorphism.

Before ending this subsection we shall briefly discuss the morphisms in CatPetri so that the reader will get a feel for what we mean by behaviour in $\mathcal{T}[N]$. We shall illustrate this by discussing refinement, as some morphisms in CatPetri can be interpreted as refinement morphisms. In the net N_1 in figure 5 we wish to refine the transition r by the net R in figure 6. A simple graphical substitution gives as a result the net N_2 in figure 7. Now there exists a morphism $f : \mathcal{T}[N_1] \to \mathcal{T}[N_2]$ in CatPetri given by the following assignments:

$$\begin{array}{rcl}
t_1 & \mapsto & {t_1}' \\
t_2 & \mapsto & {t_2}' \\
t_3 & \mapsto & {t_3}' \\
t_4 & \mapsto & {t_4}' \\
r & \mapsto & ({t_6}' \otimes {t_7}')\,;{t_8}'\ .
\end{array}$$

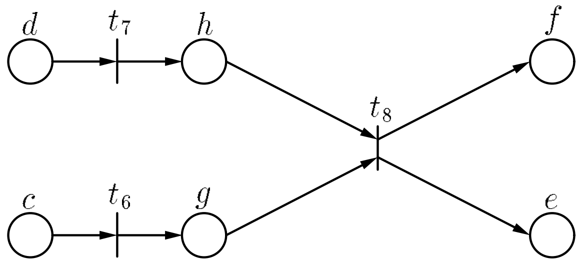


Figure 6: The net R.

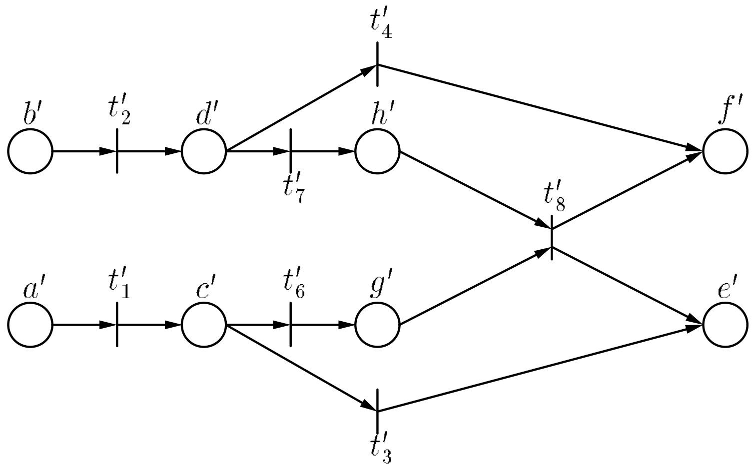


Figure 7: The net N_2.

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 $c \otimes b$ both steps ${t_3}' \otimes {t_5}' : c' \otimes d' \to f' \otimes g'$ and ${t_4}' \otimes {t_6}' : c' \otimes d' \to e' \otimes h'$ and are possible. The resulting markings are dead. The problem arises, because the net R exhibits a phenomenon called *initial concurrency*, where the initial transitions ${t_5}'$ and ${t_6}'$ need not fire concurrently. There are standard solutions to this problem (see e.g. [60] 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 behaviours 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 CatPetri it is actually mapping behaviours to behaviours. 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 $({t_6}' \otimes {t_7}')\,;{t_8}'$. A further investigation of this interpretation may be of use in the theory of refinement.

2.2 Completeness properties.

The main aim of this subsection is to show which colimits do exist in Petri. The fact that Petri does not have all colimits will lead to definition of a category PetriG which is a subcategory of Petri that has all colimits.

The lack of arbitrary limits and colimits in the category Petri follows from the lack of the corresponding limits and colimits in the category of free abelian monoids. This is illustrated by the lack of equalizers and coequalizers. The following two counter-examples are due to Josè Meseguer [51]. First equalizers:

Counterexample 1
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 2
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 1
The category <u>Petri</u> has all products.

Proof:
The net $\langle\{*\}, \phi, 0, 0\rangle$ is the terminal object. The product $N_1 \times N_2$ (see figure 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 $\pi^i = \langle\pi^i_T, \pi^i_P\rangle$ with $\pi^i_T(T_1 \times T_2) = T_i$ and $\pi^i_P(P_1 \uplus P_2) = P_i$ for $i = 1, 2$. The maps ι and o are defined by

$$\begin{aligned} \iota : \langle t_1, t_2\rangle &\mapsto \iota_1(t_1) \oplus \iota_2(t_2) \\ o : \langle t_1, t_2\rangle &\mapsto o_1(t_1) \oplus o_2(t_2)\ . \end{aligned}$$

□

Proposition 2
The category <u>Petri</u> 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 <u>Petri</u> lacks more useful colimits 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 [42]. Recall that we started with the category of graphs to which we then added structure. The choice of morphism that was made is the reason that cocompleteness was lost. By changing the morphism cocompleteness can be regained.

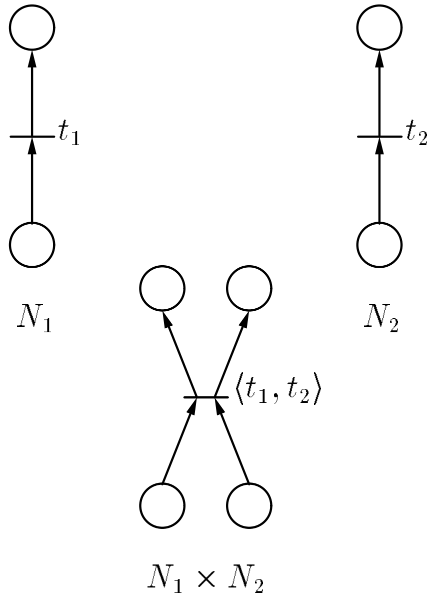


Figure 8: $N_1 \times N_2$ is the product of N_1 and N_2.

Definition 6
The category PetriG has as objects Petri nets and as morphism pairs of functions $\langle f_t, f_p \rangle$ such that

$$\begin{tikzcd} T \arrow[r, shift left, "\iota"] \arrow[r, shift right, "o"'] \arrow[d, "f_t"'] & P^{\otimes} \arrow[d, "f_p^{\otimes}"] \\ T' \arrow[r, shift left, "\iota'"] \arrow[r, shift right, "o'"'] & P^{\otimes\prime} \end{tikzcd}$$

commutes. ($f_p^{\otimes}$ is the free extension of f_p to a monoid homomorphism.)

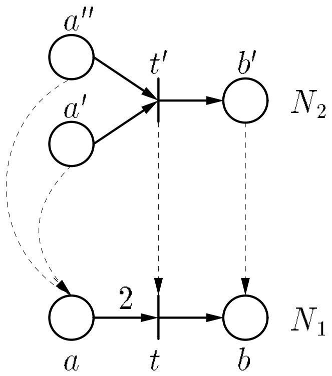
□

The following is obvious:

Proposition 3
PetriG is a wide subcategory of Petri. □

An example will illustrate the difference between the morphisms. Figure 9 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 colimits. Because the category PetriG is the subcategory of



$$\begin{aligned} f_p(a') &= a \\ f_p(a'') &= a \\ f_p(b') &= b \\ f_t(t') &= t \end{aligned}$$

Figure 9: A morphism in <u>PetriG</u>.

<u>Petri</u> that corresponds to the image of the free functor $\mathcal{F}$[<u>Graph</u>] that is left adjoint to the forgetful functor $\mathcal{U}$: <u>Petri</u> $\to$ <u>Graph</u> the category <u>PetriG</u> is cocomplete. More explicitly the constructions are show below.

Theorem 2
<u>PetriG</u> is cocomplete

Proof:
We list the simple colimits:

- 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 $N_1 \underset{g}{\overset{f}{\rightrightarrows}} N_2$ is the net $N = \langle T, P, \iota, o \rangle$ with

$$\begin{aligned} T &= coeq(T_1 \underset{g_T}{\overset{f_T}{\rightrightarrows}} T_2) \text{ in } \underline{\text{Set}} \\ P &= coeq(P_1 \underset{g_P}{\overset{f_P}{\rightrightarrows}} P_2) \text{ in } \underline{\text{Set}} \,. \end{aligned}$$

  The maps ι, o are defined by the universal property of the coequalizer.

□

Unfortunately we do not have all products in <u>PetriG</u>. Consider the nets N_1, N_2, N_3 in figure 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 <u>PetriG</u> there exist no maps $N_3 \to N_i$.

To get a category that is both complete and cocomplete we still need to make a change in the morphism on the places. The map on places is essentially a map in <u>Set</u>. If we extend the map to allow partial functions we can then construct the product. We shall not be concerned with this category in this work.

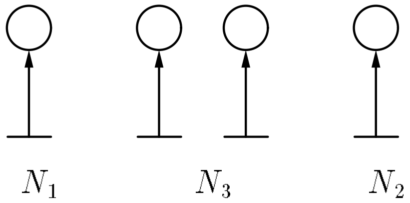


Figure 10: The category PetriG does not have all probucts.

3 Algebraic High-Level nets

The aim of this section is to define a category of algebraic high-level nets and define their semantics in terms of Petri nets. The basic idea behind algebraic high-level nets is to label the arcs of the net with terms from an algebraic specification of a datatype [22]. Because algebraic specifications have a notion of implementation given in terms of so called SPEC-algebras we are able to give a semantics for our high-level nets in terms of Petri nets. This semantics can be seen as the unfolding of the high-level net into a low-level net. The algebraic high-level nets we define are essentially the algebraic nets of Reisig and Vautherin proposed in [56].

The section is structured as follows. We start by defining algebraic high-level nets and their semantics. The main result of the first subsection is the fact that the semantics is a functor from the category of algebraic high-level nets to the category PetriG. The second subsection is concerned with the proof of cocompleteness of the category of algebraic high-level nets. This result allows us then in the third subsection to prove that the semantics functor preserves colimits. This fact is the main result of this section and it is then used in the final subsection to discuss some proof rules for compositional reasoning with high-level nets.

The results in this section are based on the results in [20] although the notion of algebraic high-level net there is slightly more complicated than ours. Our nets differ from the nets proposed by Reisig in [57] by allowing interpretations other than the initial. What this means is that we view multisets of terms essentially as an abbreaviation for multiple arcs between transitions and places, while Reisig incorporates the notion of a multiset into the specification of the abstract datatype.

We shortly define the concepts relating to algebraic specification that will be used henceforth in this work. For a more thorough discussion see [22].

An algebraic specification $SPEC = \langle SO, OP, EQ \rangle$ is composed of a set of sorts SO, a set of operator symbols OP, and equations EQ. A SPEC-algebra A contains for every sort $s \in SO$ a carrier set A_s and for every operator symbol $op \in OP$ an operation op_A, that satisfies the equations.

A homomorphism between two SPEC-algebras $f : A \to B$ is a family of maps $f_s : A_s \to B_s$ that are compatible with the operations.

The term algebra of the specification SPEC is denoted by T_{OP}. The term algebra with variables from a set X is denoted by $T_{OP}(X)$. For each SPEC-algebra A there exists an evaluation function $\mathrm{eval}_A : T_{OP} \to A$. Let $\mathrm{ass}_A : X \to A$ be a variable assignment. The $\mathrm{ass}_A^{\#} : T_{OP}(X) \to A$ denotes the unique extension of ass_A for X to $T_{OP}(X)$.

3.1 Algebraic net specifications

An Algebraic High-Level net (AHL-net) consists of three parts, an abstract datatype specification SPEC, a net that has arcs labeled with terms from this specification and a SPEC-algebra that is the semantics of the specification.

For technical reasons it is advantageous to approach the formal definition of AHL-nets in two stages. We first define an algebraic net specification.

Definition 7
An *Algebraic Net Specification* is a tuple
$NS = \langle SO, OP, EQ, X, T, P, \mathrm{pre}, \mathrm{post}, sort \rangle$ where

- $\langle SO, OP, EQ \rangle$ is a specification,
- X is a set of SO-sorted variables,
- T, P are sets of transitions and places respectively,
- $\mathrm{pre}, \mathrm{post} : T \to [P \to T_{OP}(X)^{\oplus}]$, and
- $sort : P \to SO$.

$Var(t) \subseteq X$ is the union of the variables in $\mathrm{pre}(t), \mathrm{post}(t)$. □

The definition formalises the intuitive idea that given a net we can, instead of labeling the net with integers as with P/T-systems, label the net with terms from the term algebra $T_{OP}(X)$. Actually we label the net with multisets of terms (elements of $T_{OP}(X)^{\oplus}$) because then we do not need multiple arcs between places and transitions.

We shall sometimes refer to algebraic net specifications
$NS_i = \langle T_i, P_i, \mathrm{pre}_i, \mathrm{post}_i \rangle$. In this case it is understood, that the specification part $\langle SO, OP, EQ \rangle$ is the same for all NS_i.

In figure 11 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 abstract datatype specification in the net specification has a standard notion of implementation. By now combining the net specification with an implementation, that is a SPEC-algebra, we get an algebraic high-level net.

Definition 8
Let $A \in \underline{\mathrm{Alg}(\mathrm{SPEC})}$ and NS be an algebraic net specification. Then an *Algebraic* $\overline{\textit{High-level}}$ *Net* is a pair $AN = \langle NS, A \rangle$. □

As noted previously an algebraic net specification consists of two parts, the algebraic specification part and the net part. Because the net is annotated with terms from a specification and the specification has a well defined semantics in terms of SPEC-algebras, we can ask what this semantics does to the net part of the algebraic net specification? It turns out that the semantics transforms the high-level net into a corresponding low-level net according to the following definition.

Definition 9
Given an algebraic high-level net $AN = \langle NS, A \rangle$ we can define a Petri net $F_{NS}(A) = \langle T_F, P_F, \iota_F, o_F \rangle$ as follows:

- $P_F = \bigcup_{p \in P} A_{sort(p)} \times p$,
- $T_F = \{ \langle t, \mathrm{ass}_A^{\#} \rangle \,|\, \mathrm{ass}_A \in [Var(t) \to A], t \in T \}$,
- $\iota_F(\langle t, \mathrm{ass}_A^{\#} \rangle) = \mathrm{ass}_A^{\oplus}(\mathrm{pre}(t))$,
- $o_F(\langle t, \mathrm{ass}_A^{\#} \rangle) = \mathrm{ass}_A^{\oplus}(\mathrm{post}(t))$,

with $\mathrm{ass}_A^{\oplus} : T_{OP}(X)^{\oplus} \to (P \times A)^{\oplus}$. We shall often talk of an interpretation of NS in A. By this we mean the net $F_{NS}(A)$. □

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. The construction can be viewed as an unfolding of the high-level net into a low-level net.

An example will clarify the construction. Figure 12 gives an interpretation in the algebra:

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

with

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

`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`	:	$l(ph_i) = f_i$	
		$r(ph_i) = f_i$	for $i = 1, 2$
		$r(ph_3) = f_1$	

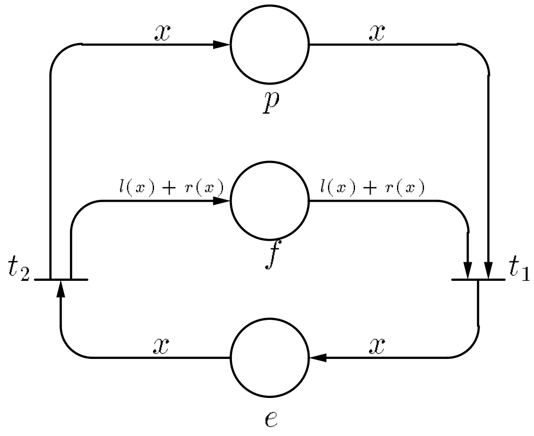


Figure 11: The specification NS of the dining philosophers problem.

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

x	r	l
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 12.

A slightly more interesting interpretation is the interpretation in the algebra $A^* = (\{*\}, \{id\})$ which gives the net in figure 13. The net shows just the amount of tokens consumed and produced by each transition, i.e. transition t_1 takes one token from place p, 2 tokens from place f, and produces one token to place e. Some uses for this interpretation are discussed in [18].

The point with this construction is, that it is functorial, because given a morphism $f : A_1 \to A_2$ with $A_1, A_2 \in \underline{\mathrm{Alg(SPEC)}}$ we have the following:

Proposition 4
Let $AN_1 = \langle NS, A_1\rangle$ and $AN_2 = \langle NS, A_2\rangle$ be algebraic high-level nets based on the same net specification. Then given a $\underline{\mathrm{Alg(SPEC)}}$-morphism $f : A_1 \to A_2$ the semantics F_{NS} can be extended to a functor $\overline{F_{NS}} : \underline{\mathrm{ADT(SPEC)}} \to \underline{\mathrm{PetriG}}$ by the following assignments:

- $F_{NS}(f)_T(\langle t, \mathrm{ass}^{\#}_{A_1}\rangle) = \langle t, \mathrm{ass}^{\#}_{A_1};f\rangle$, and
- $F_{NS}(f)_P(\langle p, a\rangle) = \langle p, f(a)\rangle$.

Proof:
We show that $F_{NS}(f)_T$ and $F_{NS}(f)_P$ form a $\underline{\mathrm{PetriG}}$-morphism, i.e.

$$\iota_{A_2}(F_{NS}(f)_T(t)) \quad = \quad F_{NS}(f)_P(\iota_{A_1}(t)),$$

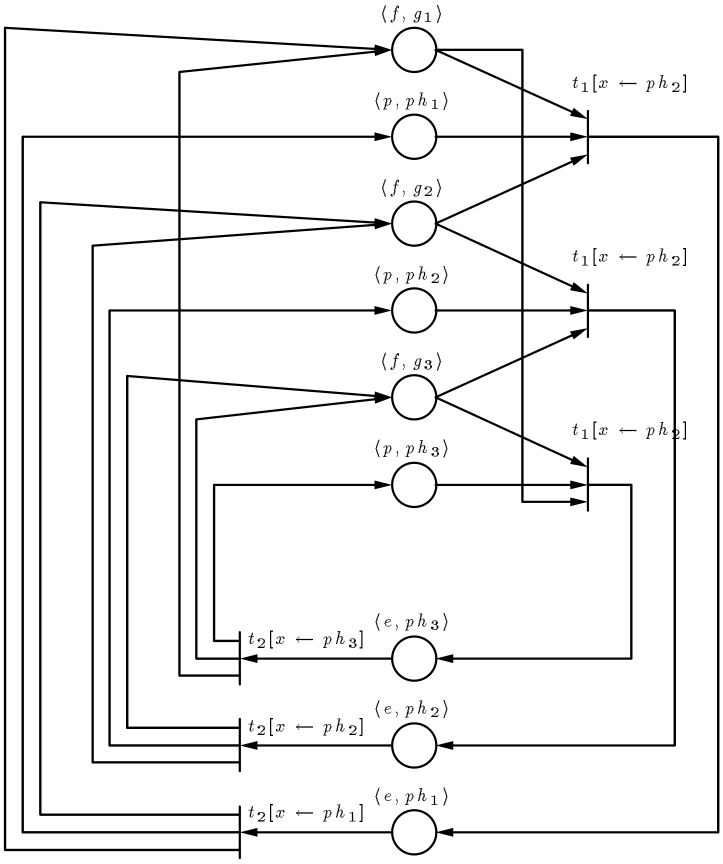


Figure 12: The interpretation of NS in A.

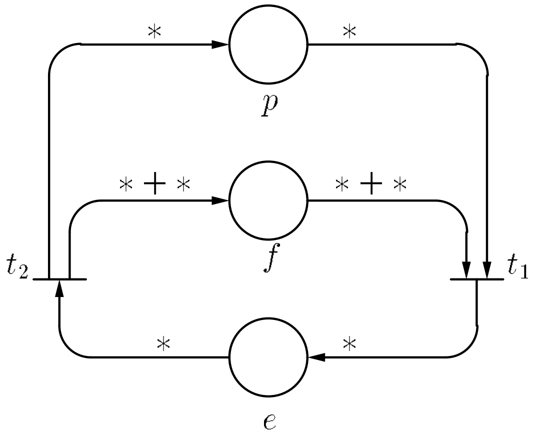


Figure 13: The interpretation of NS in A^*.

$$o_{A_2}(F_{NS}(f)_T(t)) \quad = \quad F_{NS}(f)_P(o_{A_1}(t)) \; .$$

Now for pre:

$$\begin{aligned}
\iota_{A_2}(F_{NS}(f)_T(t)) &= \iota_{A_2}(\langle t, \mathrm{ass}^{\#}_{A_1} ;f\rangle) \\
&= (\mathrm{ass}^{\#}_{A_1} ;f)^{\oplus}(\mathrm{pre}(t)) \\
&= \langle id_{P_{A_1}}, f\rangle(\mathrm{ass}^{\oplus}_{A_1}(\mathrm{pre}(t))) \\
&= F_{NS}(f)_P(\iota_{A_1}(t)) \; .
\end{aligned}$$

Analogously for post.

□

Why do we need these definitions and proofs? Because we want to define a notion of AHL-morphism. In the above definitions we have always kept the specification SPEC and the net constant. Thus we have a different functor for each net specification. In such a situation the natural category theoretic question to ask is whether given a map between net specifications there exists a natural transformation between the corresponding functors. To answer this question we need a notion of morphism between algebraic net specifications.

Definition 10
An NS-morphism between algebraic net specifications is a pair $\langle h_T, h_P\rangle : NS_1 \to NS_2$, where $h_T : T_1 \to T_2$ and $h_P : P_1 \to P_2$, such that:

- $h_P^{\S}(\mathrm{pre}_1(t)) = \mathrm{pre}_2(h_T(t))$, and
- $h_P^{\S}(\mathrm{post}_1(t)) = \mathrm{post}_2(h_T(t))$, where

$h_P^{\S} : [P_1 \to T_{OP}(X)] \to [P_2 \to T_{OP}(X)]$ is defined by

$$h_P^{\S}(M)(s) = \bigoplus_{h_p(p_1)=p,\;\; p_1 \in P_1} M(p) \; .$$

with $M \in [P_1 \to T_{OP}(X)]$ and $s \in P_2$. □

Again notice that we keep the specification constant.

Now we can prove that given an ANS-morphism there exists a natural transformation between the corresponding functors. The idea is that the functors F are indexed by the nets. Given a specification $SPEC$ each net defines a different functor F in such a way that a map between different nets induces a natural transformation between the corresponding functors.

Proposition 5
Let $h : NS_1 \to NS_2$ be an NS-morphism, then there exists a natural transformation $\alpha : F_{NS_1} \to F_{NS_2}$.

Proof:
Define $\alpha(A) = (\alpha(A)_P, \alpha(A)_T) : F_{NS_1} \to F_{NS_2}$ as follows:

$$\alpha(A)_T(\langle t, \mathrm{ass}^{\#}_A\rangle) = \langle h_T(t), \mathrm{ass}^{\#}_A\rangle,$$

and

$$\alpha(A)_P(\langle p, a\rangle) = \langle h_P(p), a\rangle\ .$$

Now for α to be a natural transformation the following equations must be satisfied:

$$\begin{array}{rcl} F_{NS_2}(f)_T(\alpha(A_1)_T(t)) & = & \alpha(A_2)_T(F_{NS_1}(f)_T(t)) \\ F_{NS_2}(f)_P(\alpha(A_1)_P(p)) & = & \alpha(A_2)_P(F_{NS_1}(f)_P(p)) \end{array}$$

The first equation is equivalent to:

$$\begin{array}{rcl} F_{NS_2}(f)_T(\langle h_T(t), \mathrm{ass}^{\#}_{A_1}\rangle) & = & \langle h(t), \mathrm{ass}^{\#}_{A_1}\, ; f\rangle \\ & = & \alpha(A_2)_T(\langle t, assA_1\, ; f\rangle) \\ & = & \alpha(A_2)_T(F_{NS_1}(f)_T(\langle t, \mathrm{ass}^{\#}_{A_1}\rangle)) \end{array}$$

For the second equation notice that:

$$F_{NS_1}(f)_P\, ; \alpha(A_2)_P = \langle id_P, f\rangle\, ; \langle h_P, id_{A_2}\rangle = \langle h_P, f\rangle$$

It remains to prove, that the compositions

$$\begin{array}{rcl} F_{NS_2}(f)_T(\alpha(A_1)_T(t)) & = & \alpha(A_2)_T(F_{NS_1}(f)_T(t)) \\ F_{NS_2}(f)_P(\alpha(A_1)_P(p)) & = & \alpha(A_2)_P(F_{NS_1}(f)_P(p)) \end{array}$$

are <u>PetriG</u>-morphisms, according to definition 6 on page 16.

$$\begin{array}{rl} & (F_{NS_1}(f)_P\, ; \alpha(A_2)_P)(\iota_{NS_1 A_1}(\langle t, ass_{A_1}\rangle)) \\ = & (F_{NS_1}(f)_P\, ; \alpha(A_2)_P)^{\oplus}(ass^{\oplus}_{A_1}(\mathrm{pre}_{NS_1}(t))) \\ = & \langle h_P, f\rangle^{\oplus}(ass^{\oplus}_{A_1}\mathrm{pre}_{NS_1}(t)) \\ = & (\mathrm{ass}^{\#}_{A_1}\, ; f)^{\oplus}(h^{\S}_P(\mathrm{pre}_{NS_1}(t))) \\ = & (\mathrm{ass}^{\#}_{A_1}\, ; f)^{\oplus}(\mathrm{pre}_{NS_2}(h_T(t))) \\ = & \iota_{NS_2 A_2}(\langle h_T(t), (\mathrm{ass}^{\#}_{A_1}\, ; f\rangle) \\ = & \iota_{NS_2 A_2}(F_{NS_2}(f)_T\, ; \alpha(A_1)_T) \end{array}$$

□

With the help of the notion of NS-morphism and SPEC-morphism we can now define the category of algebraic high-level nets.

Definition 11
The category $\underline{\text{AHL(SPEC)}}$ has as objects algebraic high-level nets, and as morphisms pairs $f_{AH} = \langle f_{ANS}, f_A \rangle : \langle NS_1, A_1 \rangle \rightarrow \langle NS_2, A_2 \rangle$ where $f_{ANS} : NS_1 \rightarrow NS_2 \in \mathbf{Mor}_{\underline{\text{ANS(SPEC)}}}$ and $f_A : A_1 \rightarrow A_2 \in \mathbf{Mor}_{\underline{\text{Alg(SPEC)}}}$.

□

The reason for the approach taken will become apparent from the ease with which we now can prove the following important fact.

Theorem 3
The assignment

$$Sem(\langle NS, A \rangle) = F_{NS}(A)$$

defines a functor

$$Sem : \underline{\text{AHL(SPEC)}} \rightarrow \underline{\text{PetriG}} .$$

Proof:
The following diagram commutes because of propositions 4,5.

$$\begin{tikzcd}
(NS_1, A_1) \arrow[r, "Sem"] \arrow[ddd, "{(f_{ANS}, f_A)}"'] & F_{NS_1}(A_1) \arrow[dd, "\alpha(A_1)"'] \arrow[dr, "F_{NS_1}(f_A)"] & \\
& & F_{NS_1}(A_2) \arrow[dd, "\alpha(A_2)"] \\
& F_{NS_2}(A_1) \arrow[dr, "F_{NS_2}(f_A)"'] & \\
(NS_2, A_2) \arrow[rr, "Sem"] & & F_{NS_2}(A_2)
\end{tikzcd}$$

□

In conclusion then the result of this section is that there exists a well defined semantics for AHL-nets in terms of Petri nets. The well definedness is expressed in the fact that there exists a functor from $\underline{\text{AHL(SPEC)}}$ to $\underline{\text{PetriG}}$. The rest of this section will be concerned with proving that the functor Sem is cocontinous and then showing some applications of this fact.

3.2 Cocompleteness

The aim of this subsection is to prove that <u>AHL(SPEC)</u> is cocomplete. The proof is approached in stages.

From the fact that <u>ANS(SPEC)</u>-morphism essentially just change the underlying net it is easy to deduce that the category <u>ANS(SPEC)</u> is cocomplete.

Theorem 4
The category <u>ANS(SPEC)</u> is cocomplete.

Proof:
We describe coproducts and coequalizers:

- *coproducts*: Let $NS_i = \langle T_i, P_i, \mathrm{pre}_i, \mathrm{post}_i, \mathrm{sort}_i \rangle$ for $i = 1, 2$. Define

  $$NS_1 + NS_2 = \langle T_1 \uplus T_2, P_1 \uplus P_2, \mathrm{pre}_1 + \mathrm{pre}_2, \mathrm{post}_1 + \mathrm{post}_2, \mathrm{sort}_1 + \mathrm{sort}_2 \rangle$$

  Clearly the injections $in_i : NS_i \to NS_1 + NS_2$ are <u>ANS(SPEC)</u>-morphisms.

  $$\begin{tikzcd} NS_1 \arrow[r, "in_1"] \arrow[dr, "f_1"'] & NS_1 + NS_2 \arrow[d, "\varphi"] & NS_2 \arrow[l, "in_2"'] \arrow[dl, "f_2"] \\ & NS & \end{tikzcd}$$

  The universal arrow $\varphi = \langle \varphi_T, \varphi_P \rangle$ is defined by $\varphi_T(t) = f_{i_T}(t)$ for $t \in T_i$ and $\varphi_P(p) = f_{i_P}(p)$ for $P \in P_i$ with $i = 1, 2$.

- *coequalizers*: Let $q_P = coeq(f_P, g_P)$ and $q_T = coeq(f_T, g_T)$ in <u>Set</u>. Define the coequalizer of

  $$\begin{tikzcd} NS_1 \arrow[r, shift left, "f"] \arrow[r, shift right, "g"'] & NS_2 \arrow[r, "q"] & NS \end{tikzcd}$$

  as

  $$NS = \langle T, P, \mathrm{pre}, \mathrm{post}, \mathrm{sort} \rangle,$$

  with $T = T_2 / \equiv_T$, $P = P_2 / \equiv_P$, $\mathrm{pre}([t])(p) = \bigoplus_{p \equiv_P p'} \mathrm{pre}_2([t])(p')$ and $\mathrm{post}([t])(p) = \bigoplus_{p \equiv_P p'} \mathrm{post}_2([t])(p)$ where $t_1 \equiv_T t_2$ iff $\exists t' \in T_1$ such that $f_T(t') = t_1$
  and $g_T(t') = t_2$; the relation $\equiv_P$ is defined analogously. It is easy to verify that

  $$\langle q_T, q_P \rangle : NS_2 \to NS$$

  is an <u>ANS(SPEC)</u>-morphism, and that it satisfies the universal property.

□

The cocompleteness of $\underline{\text{AHL(SPEC)}}$ then follows from the fact that both $\underline{\text{ADT(SPEC)}}$ and $\underline{\text{ANS(SPEC)}}$ are cocomplete. The colimits are calculated componentwise.

Theorem 5
$\underline{\text{AHL(SPEC)}}$ is cocomplete. □

3.3 Cocontinuity of Sem

The aim of this subsection is to prove the main result of this section, that the functor $Sem : \underline{\text{AHL(SPEC)}} \to \underline{\text{PetriG}}$ preserves colimits.

Again we do the proof in stages.

Proposition 6
Let NS be an algebraic net specification. Then the functor $F_{NS} : \underline{\text{ADT(SPEC)}} \to \underline{\text{PetriG}}$ preserves coproducts and coequalizers.

Proof:

- *coproduct*: Let A_1 and A_2 be SPEC-algebras. Then there clearly exists a $\underline{\text{PetriG}}$-isomorphism $F_{NS}(A_1 + A_2) \simeq F_{NS}(A_1) + F_{NS}(A_2)$, because the interpretations of A_1 and A_2 are disjoint.
- *coequalizer*: Let $NS = \langle T, P, \text{pre}, \text{post}, \text{sort}\rangle$, let $f, g : A_1 \to A_2$ be SPEC-morphisms, and let A be the coequalizer of f, g with the canonical injection $q : A_2 \to A$. Then because F_{NS} is functor we have: $F_{NS}(f) ; F_{NS}(q) = F_{NS}(g) ; F_{NS}(q)$. The universal property for $F_{NS}(q)$ follows from the universal property of q. Thus $F_{NS}(A)$ is the coequalizer of $F_{NS}(f)$ and $F_{NS}(g)$. The assignment $\text{ass}^{\#}_{A}$ is given by $\text{ass}^{\#}_{A_2} ; q$.

□

Proposition 7
Let A be a SPEC-algebra. Then:

1. if $NS_1 + NS_2$ is a coproduct in $\underline{\text{ANS(SPEC)}}$, then $F_{NS_1+NS_2}(A)$ is the coproduct of $F_{NS_1}(A)$ and $F_{NS_2}(A)$ in $\underline{\text{PetriG}}$, and
2. if NS is the coequalizer of $NS_1 \rightrightarrows NS_2$ in $\underline{\text{ANS(SPEC)}}$, then $F_{NS}(A)$ is the coequalizer of $F_{NS_1}(A) \rightrightarrows F_{NS_2}(A)$ in $\underline{\text{PetriG}}$.

Proof:

1. It is clear that the PetriG-nets $F_{NS_1+NS_2}(A)$ and $F_{NS_1}(A) + F_{NS_2}(A)$ are isomorphic, because the coproduct $NS_1 + NS_2$ is calculated on the net part.

2. Let $f, g : NS_1 \rightarrow NS_2$ be ANS(SPEC)-morphisms, and let $NS = \langle T, P, \text{pre}, \text{post} \rangle$ be the coequalizer of $NS_1 \rightrightarrows NS_2$. Because we have a natural transformation $\alpha : F_{NS_1} \rightarrow F_{NS_2}$ we have that

$$\alpha(f) \, ; \alpha(q) = \alpha(g) \, ; \alpha(q) \; .$$

The universal property is then easily verified as in the previous proof.

□

Theorem 6
The functor Sem : AHL(SPEC) → PetriG is finitely cocontinous.

Proof:
Follows from propositions 6 and 7. □

3.4 Composition with colimits

The use of colimits to combine smaller specification to larger ones is a technique first applied in the specification language CLEAR [12, 13]. It is now a standard technique in the theory of algebraic specifications. On the other hand, in categorical approaches to net-theory colimits have played little role until recently. This is maybe in part due to the negative result of Winskel in [66] on the non-existence of coproducts in the category of Place/Transition-nets with initial marking. In the paper by Meseguer and Montanari [52] limits and colimits are mentioned, but no interesting examples except the standard product and coproduct constructs are discussed. The recent papers by Dimitrovici, Hummert and Petrucci [20, 19, 21, 18, 42] contain the first thorough discussion of colimits in categories of nets. Especially the dissertation of Hummert [42] contains interesting applications of colimits to nets. He defines a notion of net module and studies the compatibility of the notion with invariants and parametrised data structures.

In the following we shall sketch some applications of our own. Specifically we will discuss how some of the standard process algebra combinators can be implemented for nets in terms of colimits. All will be implemented as push-outs. To reason about the combined net in terms of its components we have to analyze the structure of the corresponding combinators in PetriG.

The simplest and easiest combinator is the combinator for parallel composition. It is defined as the coproduct of the components. Now according to theorem 6 the interpretation of this construct is the coproduct $Sem(AN_1) + Sem(AN_2)$ in PetriG. To reason about this net in terms of its components we must study whether we can map the coproduct to the components. In PetriG this is certainly not possible, because the inverses of the injections are not PetriG-morphisms. But in $\underline{\text{Petri}_0}$ the injections have inverses. We thus have the following correspondence between transitions in the coproduct and transitions in its components.

Proposition 8
Let $AN_1 + AN_2$ be the coproduct AN_1 and AN_2 and let in_1, in_2 be the injections. Then

$$M \xrightarrow{t} M' \text{ in } AN_1 + AN_2$$

iff

$$Sem(in_1)^{-1}(Sem(M)) \to Sem(in_1)^{-1}(Sem(M')) \text{ in } Sem(AN_1)$$

and

$$Sem(in_2)^{-1}(Sem(M)) \to Sem(in_2)^{-1}(Sem(M')) \text{ in } Sem(AN_2)\ .$$

□

Truthfully it must be stated that the proposition is trivial, but the point is not so much the proposition itself as the proof technique. The idea is to find a morphism not necessarily in PetriG but in some other net-category presented in section 2.

A more interesting example is that of non-deterministic choice. Non-deterministic choice in nets raises some slightly philosophical issues that concern the autoconcurrency of transitions, i.e. the fact that a transition may be multiply enabled, that we shall shortly discuss before defining the construction. The reason for the non-existence of the coproduct in the general categories of Winskel is due, on an intuitive level, to the fact that non-determinism is difficult to interpret in the presence of non-unit weights on the input arcs.

Examine the two nets N_1 and N_2 in figure 14. They consist of only one transition each, but with net N_1 having an input weight of 2 on its transition. Suppose we want to construct the non-deterministic choice between the actions t_1 and t_2. The obvious construction is to share the input place giving the net N also in figure 14. For action t_1 to be enabled we must have an initial marking of 2 in place a. But with this marking we can perform action t_2 twice. So net N does not represent the choice $t_1 + t_2$ but the choice $t_1 + 2t_2$. The construction of the choice $t_1 + t_2$ is clearly not possible. The solution proposed by Meseguer and Montanari in [52], is to restrict the nets to have a weight of 1 on the arcs from the initially marked places. We shall also adopt

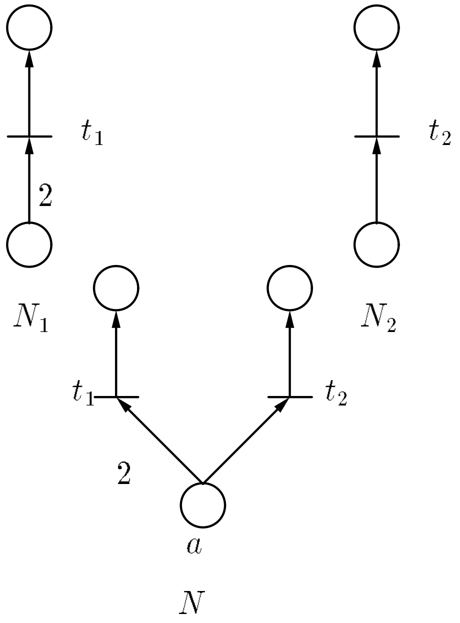


Figure 14: Nets illustrating the problem with choice.

a solution analogous to this. The nets will be of such a form, that the place on which the choice is to be made has no incoming arcs, and the outgoing arcs in both nets have the same inscription. The maps $Sem(in_i)^{-1}$ are now constructed in CMonPetri. All the places of the other net are mapped onto the place s and the transitions on the idle transition t_s. Then we have:

Proposition 9
Let AN_1 and AN_2 be two AHL-nets. Denote the place on which the choice is to be made by s. Define a net S consisting of only the place s. The non-deterministic choice between the nets is now the push-out $AN = AN_1 \leftarrow S \rightarrow AN_2$. Let M be the marking with only place s occupied. Then

$$M \xrightarrow{t} M' \text{ in } AN$$

iff

$$Sem(in_1)^{-1}(Sem(M)) \rightarrow Sem(in_1)^{-1}(Sem(M')) \text{ in } Sem(AN_1)$$

or

$$Sem(in_2)^{-1}(Sem(M)) \rightarrow Sem(in_2)^{-1}(Sem(M')) \text{ in } Sem(AN_2)\ .$$

□

The proof of the proposition is easy, because $Sem(in_i)^{-1}$, being a CMonPetri-morphism preserves behaviour.

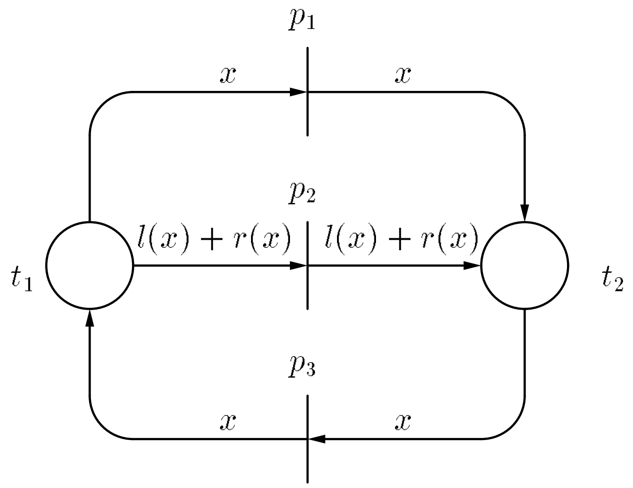


Figure 15: The dual of the dining philosophers net.

Sequential composition of two nets can also be constructed as a push-out. Given two nets AN_1 and AN_2 that we want to compose, we need to specify the places on which they are to be patched together. Let this set be IO. Construct the net with only places from IO. The sequential composition of AN_1 and AN_2 is now the pushout $AN = AN_1 \leftarrow IO \rightarrow AN_2$. The behaviour of the composition is easily deduced from the behaviour of the components.

The last notion typically found in process calculi is that of synchronisation. This is where push-outs fail to give a nice characterisation. The reasons are rather technical, but the reader can get an intuition for the problem by recalling that both Winskel and Meseguer and Montanari construct synchronisation as a product, i.e. a limit not a colimit. Because synchronisation is the sharing of transitions between nets, the implementation in terms of push-outs would require the existence of a map from a net with a single transition and no places to some other net. But because maps are supposed to preserve the environment of a transition, and this is not the case for the required map, such a map does not exist.

Synchronisation can be implemented be means of push-outs, but it requires the introduction of a new notion of net duality. The dual of a net N is the net N^{op} that has as transitions the places of N and as places the transitions of N. Figure 15 depicts the dual of the philosophers net in figure 11.

Synchronisation can now be implemented by taking the dual of the nets to be synchronised, constructing the push-out and thereby sharing the transitions (which now are places) and taking the dual again. The steps are illustrated in figures 16-19. With respect to the behaviour we can now say the following:

Proposition 10
Let AN be the synchronisation of the nets AN_1 and AN_2 constructed as

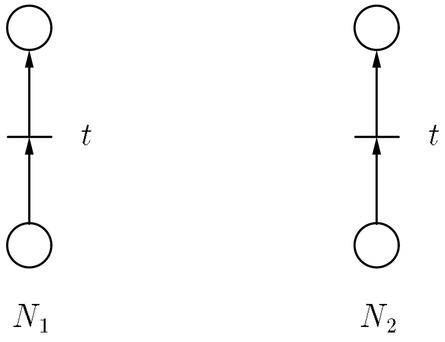


Figure 16: The nets to be synchronised at transitions t.

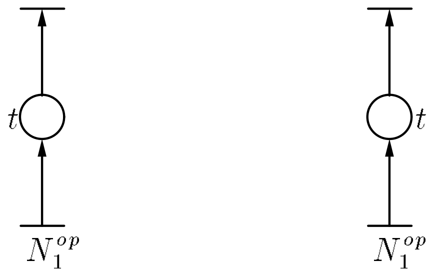


Figure 17: The duals of the nets above.

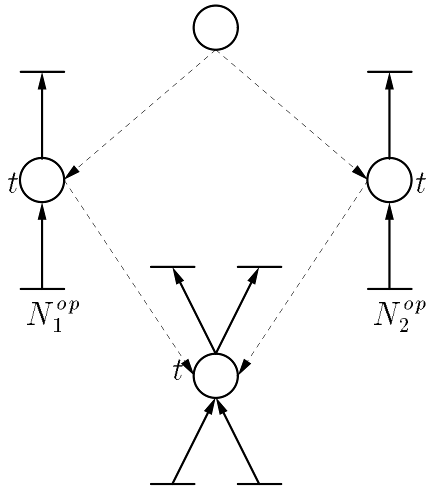


Figure 18: The push-out construction.

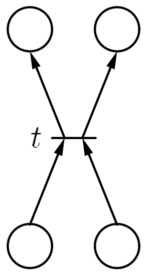


Figure 19: The result of taking the dual of the push-out.

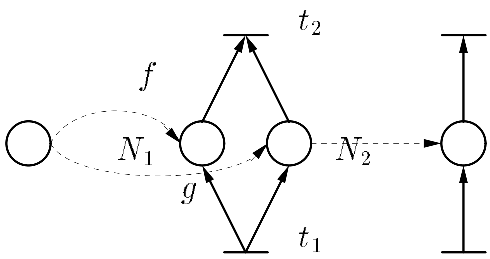


Figure 20: A net-reduction considered as a coequalizer.

described above. Then

$$M \xrightarrow{t} M' \text{ in } AN$$

iff

$$Sem(pr_1)(Sem(M)) \to Sem(pr_1)(Sem(M')) \text{ in } Sem(AN_1)$$

and

$$Sem(pr_2)(Sem(M)) \to Sem(pr_2)(Sem(M')) \text{ in } Sem(AN_2)\ .$$

where the projections pr_i are defined as:

$$pr_{1T}(t) = \begin{cases} t & \text{iff } t \in T_1 \\ 0 & \text{iff } t \in T_2, \end{cases}$$

and

$$pr_{1P}(p) = \begin{cases} p & \text{iff } p \in P_1 \\ 0 & \text{iff } p \in P_2, \end{cases}$$

in $\underline{\text{Petri}_0}$. The definition of pr_2 is analogous. □

The proof is again easy because pr_1 and pr_2 are $\underline{\text{Petri}_0}$-morphisms, and thus preserve reachable markings.

As a last example we shall shortly evaluate the application of colimits to describe net transformations and specifically net reductions (cf. [6]). By a net-reduction we shall mean a transformation of a net into a simpler but equivalent net. For an example consider the net N_1 in figure 20. The existence of the two places between the transitions t_1 and t_2 seems superfluous, because the net N_2 in the same figure has an equivalent behaviour. The reduction can be constructed as a coequalizer as shown in the figure. This is a very simple reduction. More complex reductions like the reduction of sequential chains of transitions to one single transition seem to require a more complex notion of net-morphism allowing for change of signature and the mapping of operators in one signature to derived operators in an other. This would seem a profitable area to further research.

In this section we have defined a category of high-level nets, and given its semantics by a functor into the category PetriG. In the next section this functor will be exploited in the construction of a model for linear logic.

4 Linear Logic and AHL-Nets

The aim of this section is to show how linear logic can be used to describe properties of AHL-nets. It is structured as follows. The first subsection gives a short introduction to linear logic. In the second subsection a class of models for linear logic called quantales are presented. It is shown how a Petri net generates a quantale. In the third and last subsection we present the interpretation of linear logic in quantales and discuss how properties of AHL-nets are expressed.

4.1 What is linear logic?

Linear logic was discovered by J-Y. Girard [28] while trying to extend his coherent semantics, a domain theoretic model of second-order lambda-calculus [35], 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 [31], from the linear algebra like nature of some of the connectives of linear logic.

Modern introductions to linear logic like appendix B in [36] introduce linear logic through the sequent calculus; as this to date is the most accessible way to understanding linear logic. Essentially the different flavours of linear logic (intuitionistic, classical and predicate) are obtained by deleting the contraction and weakening rules from the standard sequent calculus formulations of the corresponding logics. In figure 21 the sequent calculus formalisation of linear intuitionistic predicate logic is given.

When comparing the formulation in figure 21 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 give intuitive interpretations in terms of the behaviour of a net to all these connectives in subsection 4.3. 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

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:

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

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

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

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

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

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

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

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

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

Figure 21: Sequent calculus for intuitionistic predicate LL.

and $\wedge$ connective in the sequent calculus of intuitionistic logic:

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

The correctness of this rule is based on the interpretation of a sequent $\Lambda \vdash C$ as *the conjunction of all the premisses 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. [25]). 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 21. 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 21 by the rule ($\otimes$-R). 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 21 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 $A^{\perp} = A \multimap \perp$.

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 [28]. They all have different properties in the sense that they enlighten different aspects about the provability in intuitionistic logic.

The remaining connectives are the quantifiers. Girard gives their definition in semantic terms as infinite generalisations of the additives. Their names are *any* $\bigvee$ and *some* $\bigwedge$. Because they are defined as infinite additives they are less general than the classical connectives. The quantifiers have been studied by Girard in [29], but there only their proof-theoretic properties are investigated.

The above discussion was on a very intuitive level and was meant to give the reader a feel for linear logic. The reader interested in more details is referred to the book [36] and the paper [28]. For the reader fluent in french there exists a very readable paper by Girard published in the french version of Scientific American [33]. Finally for the reader wishing just to get a glimpse of the subject the short paper by A. Scedrov [58] is highly recommended.

4.2 Quantales and nets

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

Quantales were originally introduced by Mulvey [54] in an attempt to cast light on the connections between C^*-algebras and quantum mechanics. We do not claim to understand what this means. But the work by Mulvey and Girard's attempts to give a semantics of linear logic in terms of C^*-algebras [30, 32] inspired several authors to study quantales and their relation to linear logic [2, 69]. In this subsection we shall show how given a net we can construct a quantale that describes the behaviour. The construction described in this section was independently discovered by Brown [8] and Engberg and Winskel [24]. The general idea is to view places as propositions. The interpretation of a proposition p will then be the set of all markings that are reachable from the corresponding place p in the net [8] or, the set of all markings that are a prerequisite for the fact that place p become marked [24]. The construction is based on the facts that the reachability relation defines a quantic nucleus (a map that is weaker than a quantale morphsim) and that the image of a quantic nucleus is a quantale. This allows for a very elegant formalisation of the construction in two steps. First a quantale that is not related to the behaviour of the net is generated. Secondly the reachability relation is used to generate a quantic nucleus which then gives us the wanted quantale. Our contribution is the proof that both constructions yield a functor from the category $\underline{\text{PetriG}}$ to the category of quantales $\underline{\text{Quant}}$. We can then use the functor from $\underline{\text{AHL(SPEC)}}$ to $\underline{\text{PetriG}}$ to extend the interpretation to high-level nets. The exact details of the interpretation are described in section 4.3.

The subsection first presents the relevant background on quantales, and then discusses the details of the construction and proves its functoriality. Finally a variation of the construction is presented.

4.2.1 Quantales

The aim of this subsubsection is to give the relevant background that allows us to state theorem 7, which tells us that the image of a quantic nucleus is a quantale.

A quantale is essentially a complete lattice enriched with a monoidal operation. For the relevant notions of lattice theory and the theory of ordered sets we refer to [45] and [62].

Definition 12
A commutative quantale Q is a quadruple $\langle Q, \leq, \mathbf{I}, \otimes \rangle$ such that:

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

□

Although a complete join semi-lattice is actually a complete lattice (cf. [45] I.4.4), we want to make this distinction because we want a loser notion of morphism than that of a lattice homomorphism.

Definition 13
A morphism of quantales is a function $f : Q_1 \to Q_2$ that is monotonic and preserves $\bigvee$, $\otimes$, and $\mathbf{I}$. □

Proposition 11
Quantales and their morphism form a category Quant. □

As previously mentioned the quantale representing the net will be constructed in two steps. The first step will construct a quantale from a net, but this quantale will not represent the behaviour of the net. This quantale is then transformed into a quantale describing the behaviour of the net in the second step. The idea in the second step is to take sets of markings closed under forward or backward reachability as the elements, and prove that this construction gives a quantic nucleus. Then we can use theorem 7 and get the wanted quantale.

To formalise this we need some technical definitions and lemmata. To prove the correctness of the construction we need to show that taking the forwards or backwards closure under reachability corresponds to what is called a quantic nucleus. To define a quantic nucleus we first need the definition of a closure operator.

Definition 14
Let Q be a quantale. A function $j : Q \to Q$ is a *closure operator* iff

1. $a \leq b \Rightarrow j(a) \leq j(b)$ (j is monotonic),
2. $a \leq j(a)$ (j is increasing), and
3. $j(a) = j(j(a))$ (j is idempotent).

□

A quantic nucleus is a closure operator that loosely respects the monoidal operation of the quantale.

Definition 15
A closure operator $j : Q \to Q$ is a *quantic nucleus* iff

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

□

The following lemmata will be of use in subsection 4.2.3.

Lemma 1
$a \leq b \Rightarrow a \otimes c \leq b \otimes c$ for all $c \in Q$.

Proof:

$$\begin{aligned} a \leq b \Rightarrow (a \vee b) \otimes c &= b \otimes c \\ \Leftrightarrow (a \otimes c) \vee (b \otimes c) &= b \otimes c \\ \Leftrightarrow a \otimes c &\leq b \otimes c\ . \end{aligned}$$

□

Lemma 2
$j(a \otimes b) = j(a \otimes j(b)) = j(j(a) \otimes b) = j(j(a) \otimes j(b))$.

Proof:

$$\begin{aligned} a &\leq j(a) \\ \Rightarrow a \otimes b &\leq j(a) \otimes b \\ \Rightarrow j(a \otimes b) &\leq j(j(a) \otimes b) \;. \end{aligned}$$

$$\begin{aligned} b &\leq j(b) \\ \Rightarrow j(a) \otimes b &\leq j(a) \otimes j(b) \\ \Rightarrow j(j(a) \otimes b) &\leq j(j(a) \otimes j(b)) \;. \end{aligned}$$

On the other hand from the definition of a quantic nucleus we have

$$\begin{aligned} j(a) \otimes j(b) &\leq j(a \otimes b) \\ \Rightarrow j(j(a) \otimes j(b)) &\leq j(j(a \otimes b)) \\ \Rightarrow j(j(a) \otimes j(b)) &\leq j(a \otimes b) \;. \end{aligned}$$

Combining the above we get

$$j(a \otimes b) = j(a \otimes j(b)) = j(j(a) \otimes b) = j(j(a) \otimes j(b)) \;.$$

□

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 [55] tells us that a quantic nucleus is a quantale morphism, although not an endomorphism.

Theorem 7
Let Q be a quantale and $j : Q \to Q$ a quantic nucleus. Then the image of 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. □

4.2.2 The construction

In this subsection we shall show how the reachability relation in a net defines a quantic nucleus. Take the set of all markings of a net $mark(N)$. As previously seen the markings of a 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 $P, Q \subseteq \mathit{mark}(N)$ define

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

The following two lemmata are obvious.

Lemma 3
Let $N = \langle T, P, \iota, o \rangle$ be a net. Let $\mathcal{P}(\mathit{mark}(N))$ be the powerset of the markings of N. Let $\underline{0}$ be the zero marking ie. the unit of the monoid P^+. Then $\langle \mathcal{P}(\mathit{mark}(N)), +, \varnothing \rangle$ is a commutative monoid. □

Lemma 4
$\langle \mathcal{P}(\mathit{mark}(N)), \subseteq, +, \varnothing \rangle$ is a commutative quantale. Furthermore the assignment

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

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

The use of quantic nuclei depends heavily on the linearity and monotonicity of the behaviour of Petri nets.

Lemma 5
Let m_1, m_1', m_2, m_2' be markings of a net N, such that $m_1 \leq m_2$ and $m_1' \leq m_2'$. Then

$$m_1 + m_1' \leq m_2 + m_2' .$$

□

In the sequel the ordering $\leq$ on the markings is to be interpreted in terms of the reachability relation as follows:

$$m_1 \leq m_2 \quad \text{iff} \quad m_2 \rightarrow m_1,$$

ie. $m_1 \leq m_2$ iff m_1 is reachable from m_2. The ordering is easily extended to sets of markings:

Definition 16
Let A, B be sets of markings. Then $A \leq B$ iff for every $a \in A$ there exists $b \in B$ such that $a \leq b$. □

Given an ordering we can always define upward and downward closure operators. In the sequel we shall focus on downward closure. We shall return to upward closure in subsection 4.2.4.

Definition 17
Let N be a net. For $A \subseteq mark(N)$ the *forward evolution* of A is the endofunction:

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

□

It is obvious that

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

when A, B are downward closed.

Proposition 12
$\downarrow : q(N) \rightarrow q(N)$ is a quantic nucleus.

Proof:
Clearly $\downarrow$ is increasing and idempotent. For the other properties:

$$\begin{array}{rcl} A \leq B & \Rightarrow & A \leq \downarrow(B) \\ & \Rightarrow & \downarrow(A) \leq \downarrow(\downarrow(B)) \\ & \Rightarrow & \downarrow(A) \leq \downarrow(B) \ . \\ \downarrow(A) + \downarrow(B) & = & \{m | \exists a \in A : m \leq a\} \\ & & +\{m | \exists b \in B : m \leq b\} \\ & \subseteq & \{p | \exists m \in (A+B) : p \leq m\} \\ & = & \downarrow(A+B) \ . \end{array}$$

□

The results of this subsection are summed up in the following theorem.

Theorem 8
The following data define a quantale:

- the elements are subsets of $mark(N)$ closed under forward evolution,
- the ordering is subset inclusion,
- the monoidal operation is $A \otimes B = \downarrow(A+B)$, and
- the unit is $I = \downarrow(\underline{0})$.

□

4.2.3 The functoriality of $\downarrow$

In this subsection we shall prove 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 we can only claim the following:

Lemma 6
Let $f : N_1 \to N_2$ be a net morphism. Then $f(A) \subseteq \downarrow^{N_2}(f(A))$. $\square$

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 9
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. The the assignment $\downarrow(f)(A) = \downarrow^{N_2}(f(A))$ for $A \in \downarrow q(N_1)$ 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.

1. $\downarrow(f)$ is monotonic:

$$\begin{aligned}
A \leq B &\Rightarrow A \subseteq B \\
&\Rightarrow f(A) \subseteq f(B) \quad f \text{ preserves reachability} \\
&\Rightarrow \overset{N_2}{\downarrow}(f(A)) \subseteq \overset{N_2}{\downarrow}(f(B)) \\
&\Rightarrow \overset{N_2}{\downarrow}(f(A)) \leq \overset{N_2}{\downarrow}(f(B)) \\
&\Rightarrow \downarrow(f)(A) \leq \downarrow(f)(B)\,.
\end{aligned}$$

2. $\downarrow(f)$ preserves joins:

$$\begin{aligned}
\downarrow(f)(A \vee B) &= \overset{N_2}{\downarrow}(f(A \vee B)) \\
&= \overset{N_2}{\downarrow}(f(A) \vee f(B)) \\
&= \overset{N_2}{\downarrow}(f(A)) \vee \overset{N_2}{\downarrow}(f(B)) \\
&= \downarrow(f)(A) \vee \downarrow(f)(B)\,.
\end{aligned}$$

3. $\downarrow(f)$ preserves the tensor:

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

Thus $\downarrow(f)(A \otimes_{N_1} B) = \downarrow^{N_2}(f(A)+f(B)))$. On the other hand by lemma 2:

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

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:

$$\begin{array}{rcl}
\downarrow(f)(\mathbf{I}_{N_1}) & = & \overset{N_2}{\downarrow}(f(\overset{N_1}{\downarrow}(\underline{0}))) \\
& \subseteq & \overset{N_2}{\downarrow}(\overset{N_2}{\downarrow}(f(\underline{0}))) \\
& = & \overset{N_2}{\downarrow}(\underline{0}) \\
& = & \mathbf{I}_{N_2} \ .
\end{array}$$

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

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

$\square$

4.2.4 Upwards closed sets of markings

The choice for the ordering $\leq$ is arbitrary and other possibilities exist. For example Engberg and Winskel [24] 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. The verification that the assignment $\uparrow(f)(A) = \uparrow(f(A))$ defines a functor $\uparrow : \underline{\text{PetriG}} \to \underline{\text{Quant}}$ is also routine

We should now ask ourselves what the differences between the quantales based on upward and downward closed sets of markings are. 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. The technical differences are more straightforward except for the treatment of linear implication, but we shall return to this below.

4.3 The interpretation of Linear Logic in AHL-Nets

In this subsection we will show how linear logic can be used to express properties of AHL-nets. To talk about properties of the net we introduce typed intuitionistic predicate linear logic. The introduction of a typed version of intuitionistic linear predicate logic is strictly speaking not necessary, but it simplifies the generation of a language from a net. We shall not pay much attention to the types as there are standard methods for converting typed predicate logics into untyped variants (cf. [23]). The development of the language follows [38].

Definition 18
A *first order signature* Ω is a triple $\Omega = \langle S, \Sigma, \Pi \rangle$ where:

- S is a set of *sorts*,
- Σ is an $S^* \times S$-indexed family of sets of *operator* or *function* symbols, and

- Π is an S^*-indexed family of sets of *predicate* or *relation* symbols.

$\square$

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 19
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)-*formula*

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

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 $Free$ 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. $Var(A \square B) = Var(A) \cup Var(B)$, and $Free(A \square B) = Free(A) \cup Free(B)$ with $\square \in \{\otimes, \oplus, \&, \multimap\}$,

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 the typed linear intuitionistic predicate calculus is given in figure 21.

To formulate properties of an AHL-net in a language L_Ω we must define how the AHL-net generates a language.

Definition 20
The first order signature Ω_{AN} generated by an AHL-net AN is given by

1. $S = SO$,
2. $\Sigma = OP$, 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 $\downarrow(Sem(NS, A))$ defines a quantale Q_{AN} for the net AN. The language L_{AN} can now be interpreted in the quantale Q_{AN}.

Definition 21
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. $[\![(\bigvee x)P]\!] = \bigcap_{a \in A} [\![P[x/a]]\!]$, and
10. $[\![(\bigwedge x)P]\!] = \bigcup_{a \in A} [\![P[x/a]]\!]$.

□

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 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 place it denotes, in other words the set of resources we could obtain if we had π. The connectives are then interpreted as follows. Anything possible to gain from a marking in $[\![A \otimes B]\!]$ must be possible to gain by having both resources $a \in [\![A]\!]$ and $b \in [\![B]\!]$. Then if x is a resource we gained by a non-deterministic choice between resources A and B, it must be a consequence of either A or B. Thus $A \oplus B$ is the union of the consequences of A and B. On the other hand if we have a consequence x of $A\&B$, we know that regardless of our choice of A or B x will always be a consequence of A or B respectively. It is then natural to interpret & as disjoint union. 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 $b \in [\![B]\!]$. 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 assignment $\mathrm{ass}_A^{\#^x}$. The interpretation of the $\bigwedge$-quantifier (some) is analogous.

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 **I**.

Definition 22
An Ω-sentence P is true in a quantale Q_{AN} (ie. $Q_{AN} \models P$)

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

□

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

Lemma 7
$\models m \multimap m'$ iff marking m is reachable from m'. $\square$

When taking the interpretation proposed by Engberg and Winskel the direction of the linear implication in the lemma is changed, i.e the lemma becomes $\models m \multimap m'$ iff marking m' is reachable from m.

Proposition 13
The interpretation as given above is sound with respect to the sequent calculus,

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

Proof:
By case inspection and induction over the length of the sequents. For example the proof soundness with respect to the rule for existential quantifiers is as follows.
By hypothesis we have $\Lambda \vdash A[t/x]$, that is $[\![\Lambda]\!] \subseteq [\![A[t/x]]\!]$. Now by definition 21 we have $[\![A[t/x]]\!] \subseteq [\![(\bigwedge x)A]\!]$, which by transitivity implies $[\![\Lambda]\!] \subseteq [\![(\bigwedge x)A]\!]$, i.e. $\Lambda \vdash (\bigwedge x)A$. $\square$

Some examples will illustrate the definitions.

Example 1
Let AN be the dining philosophers net of figure 11 on page 22 with the interpretation of figure 12. Then we can state the following properties:

1. A philosopher 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

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

2. 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

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

while

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

so that

$$[\![C]\!] = \{ \underline{0}, \langle f, g_2 \rangle, \langle f, g_1 \rangle \}$$

and thus

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

3. 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

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

4. 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.

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

Now,

$$\llbracket p(ph_i) \otimes f(g_1) \otimes f(g_2) \otimes f(g_3) \rrbracket = \\ \{\langle p, ph_i\rangle + \langle f, g_1\rangle + \langle f, g_2\rangle + \langle f, g_3\rangle, \langle e, ph_i\rangle + \langle f, g_{i+2 \bmod 3}\rangle\},$$

and

$$\llbracket e(ph_i) \otimes f(g_j) \rrbracket = \\ \{\langle p, ph_i\rangle + \langle f, g_i\rangle + \langle f, g_{i+1 \bmod 3}\rangle + \langle f, g_j\rangle, \langle e, ph_i\rangle + \langle f, g_j\rangle\} .$$

One can now see, that

$$\underline{0} \in \llbracket e(ph_i) \otimes f(g_j) \multimap p(ph_i) \otimes f(g_1) \otimes f(g_2) \otimes f(g_3) \rrbracket$$

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

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

5. A property describing transition t_1:

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

because

$$\llbracket p(x) \otimes f(l(x)) \otimes f(r(x)) \rrbracket = \\ \bigcup_{i \in \{1,2,3\}} \{\langle p, ph_i\rangle + \langle f, g_i\rangle + \langle f, g_{i+1 \bmod 3}\rangle, \langle e, ph_i\rangle\},$$

and thus

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

As we can see, the calculation of the interpretation of the quantifiers is very complicated. But this is was is to be expected when working with quantifiers. □

Notice also the form of the lhs of the implication in example 1.4. 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 8
$\llbracket m \otimes 1 \rrbracket = \downarrow\{m' | m' \leq m\}$.

Proof:

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

□

Example 1.4 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 section we presented a construction for generating a quantale from a net. The functoriality of the construction was then proved. Finally using this construction and the results from the previous chapter an interpretation of intuitionistic predicate linear logic in terms of algebraic high-level nets was proposed.

5 Conclusions

In this thesis we have presented a high-level net class called algebraic high-level nets, discussed some compositionality properties of this class and proposed a language for describing properties of these nets. In this section we shall first give a brief review of the contents of this thesis, then mention some questions that we left open, discuss the connections with other high-level formalisms and finally give some suggestions on topics for further research.

We started by describing different categories of Petri nets. The simplest of these categories Petri has Petri nets without capacities as objects. The intention was to use these as semantics for high-level nets. To this end we studied the different kinds of morphisms of these categories and their completeness and cocompleteness properties. Finally we defined a category PetriG that has all colimits.

Then we defined algebraic high-level nets. These nets are essentially a generalisation of P/T-nets with the arc-weights replaced by terms from an algebraic theory. The semantics of these nets was defined in terms of a functor from the category of algebraic high-level nets the category of Petri nets PetriG. This functor was then proved to be cocontinuous, that is, to be colimit preserving. Using this functor we proposed some rules for compositional reasoning with high-level nets. The correctness proofs of these rules made good use of the hierarchy of net-categories defined in section 2. It should be noted that rules for compositional reasoning with nets have also been presented by Winskel in [66]. The novelty in our approach is that we give the rules for high-level nets and the rules are all based on push-outs.

To be able to express properties of these nets we introduced linear logic. The models of linear logic are quantales. We showed how to construct a quantale from a net, such that the quantale reflects the behaviour of the net. This construction was proved to be functorial. By composing this functor with the semantics functor from section 3 the interpretation of linear logic in net-quantales was extended to predicate linear logic. The interpretation allows one to express propositions about the reachability of markings. Several examples were discussed.

There are several questions that we did not touch upon. Even though the problem of synchronisation may be solved with a different kind of morphism, the exact properties of the duality construction for high-level nets are not clear and should be investigated. Also the preservation of limits and colimits by the functors $\downarrow, \uparrow$ is an interesting question that merits further study. On a more philosophical level there is the problem of the intuitive interpretation of the additives $\&, \oplus$. The usual interpretation is that they describe different aspects of choice, namely "internal" and "external" choice, the idea being that with internal choice the environment is unable to influence the choice, while with external choice, the environment is specifically able to force one or the other of the behaviours. But, as pointed out by Brown in [8], a fundamental assumption in net theory is that no external observer can influence the course of events, and that conflicts are resolved internally by the net. So the exact meaning of these connectives is still unclear. It should be noted that the aim of this work was to get an interpretation of linear predicate logic, so an answer to this problem was not expected.

The most known high-level net formalisms are Predicate/Transition nets (PrT-nets) [26] and Coloured nets [44]. PrT-nets arise through the idea of "dynamising" predicate logic, using predicates with changing extensions. Due to the use of variables, formal expressions, products and sums, PrT-nets are essentially a syntax based formalism. On the other hand Coloured nets can be seen as a more semantically oriented model, because they have been motivated as shorthands for conventional Petri nets. Our algebraic high-level nets can be seen as a mixture of both the previous models. As in PrT-nets we have variables, formal expressions and sums, but for example our formalism lacks the ability to add conditions on transitions. On the other hand we have sums of functions (operators) as in Coloured nets. An interesting point is also that given our interpretation of linear predicate logic, the predicates can be understood in terms of the "dynamic predicates" of PrT-nets.

Both the theory of PrT-nets and the theory of Coloured nets discuss the topic of invariant analysis very thoroughly. It is not known what kind of invariants our algebraic high-level nets posses. We conjecture that they would not differ very much from the invariants discussed by Reisig and Vautherin [61], as our nets differ only slightly from those of Reisig and Vautherin.

Neither PrT-nets nor Coulured nets have a notion of morphism. This should not be taken as criticisms against these formalisms, as for example there exists a large number of equivalence transformations for PrT-nets that achieve some of the aims of section 3 much better. On the other hand the lack of morphism means that compositional design methods have to be based on other ideas. For Coloured nets there exists an extension called Hierachical Coloured nets [41]. It is not clear how such a concept could be integrated into our formalism through categorical methods. On the other hand by defining a notion of morphism for PrT-nets and Coloured nets the ideas presented here should transfer easily.

This work opens up several paths for further research. At the end of section 3, a small example with a net-reduction is discussed. In this example it is seen, that the notion of morphism between AHL-nets is not strong enough to express more complex reductions. First of all it suffers from the same deficiency as its low-level counterpart, namely it cannot express synchronisation of transitions. Secondly net-reductions for high-level nets achieve the reduction by folding the net into the annotations. Thus it should be possible to map terms into derived terms and even change signatures. So one future area of research would be the definition of a more complex notion of morphism of AHL-nets and a systematic study of net-reductions with this new notion of morphism.

In section 4 the interpretation given to predicates is that they are places of the net. This is a natural interpretation when we are talking about unary predicates because we can think that $P(a)$ is true iff token a is in place P. But what do non unary predicates mean? Also the logic as such is rather unexpressive, because when working with practical applications one often is interested in sequences of transitions. These are not expressible in the logic. From a proof-theoretic point of view this means that we want to examine the proof of the proposition. This is one of the advantages with the use of dynamic logic in the analysis of nets as proposed by Tuominen [59].

References

[1] Abadi, M. and Plotkin, G. D. *A logical view of composition and refinement (excerpt).* Third Workshop on Concurrency and Compositionality, Goslar. 1991, pp. 1–10.

[2] Abramsky, S. *Linear process logic.* Notes by Steven Vickers, 1988.

[3] Abramsky, S. and Vickers, S. *Quantales, observational logic, and process semantics.* Research Report DOC 90/1. Department of Computing, Imperial College, London, England, 1990.

[4] Andreoli, J.-M. and Pareschi, R. *Linear objects: Logical processes with built-in inheritance.* 7th International Conference on Logic Programming, Jerusalem. May 1990.

[5] Asperti, A., Gorrieri, R., and Ferrari, G. *Implicative formulae in the state-as-proposition analogy.* Conference on Principles of Programming Languages, 1990.

[6] Berthelot, G. *Transformations and decompositions of nets.* Petri Nets: Central Models and Their Properties. Lecture Notes in Computer Science 254. Springer-Verlag, Berlin, 1986, pp. 359–376.

[7] Best, E. and Devillers, R. *Sequential and concurrent behaviour in Petri net theory.* Theoretical Computer Science, 55(1987), pp. 87–136.

[8] Brown, C. *Linear logic and Petri nets: Categories, algebra and proof.* PhD thesis, Department of Computer Science, University of Edinburgh, Scotland, 1991.

[9] Brown, C. *Petri nets as quantales.* Technical Report ECS-LFCS-89-96. Laboratory for Foundations of Computer Science, University of Edinburgh, Scotland, 1989.

[10] Brown, C. *Relating Petri nets to formulae of linear logic.* Technical Report ECS-LFCS-89-87. Laboratory for Foundations of Computer Science, University of Edinburgh, Scotland, 1989.

[11] Brown, C. and Gurr, D. *A categorical linear framework for Petri nets.* 5th Symposium on Logic in Computer Science. IEEE Computer Press, 1990.

[12] Burstall, R. and Goguen, J. A. *Putting theories together to make specifications.* 5th International Joint Conference on Artificial Intelligence. Department of Computer Science, Carnegie-Mellon University, USA, 1977, pp. 1045–1068.

[13] Burstall, R. and Goguen, J. A. *The semantics of Clear, a specification language*. Proceedings of 1979 Copenhagen Winter School on Abstract Software Specification. Lecture Notes in Computer Science 86. Springer-Verlag, Berlin, 1990, pp. 292–332.

[14] Cerrito, S. *A linear semantics for allowed logic programs*. 5th IEEE Symp. on Logic in Computer Science. IEEE Press, 1990.

[15] de Paiva, V. C. *The dialectica categories*. AMS Conference on Categories in Computer Science 1987, Boulder. Contemporary Mathematics 92. American Mathematical Society, Providence, Rhode Island, USA, 1989.

[16] de Paiva, V. C. *Dialectica-like model of linear logic*. Category Theory and Computer Science. Lecture Notes in Computer Science 389. Springer Verlag, Berlin, 1989, pp. 341–356.

[17] Degano, P., Meseguer, J., and Montanari, U. *Axiomatizing net computations and processes*. 4th Symposium on Logic in Computer Science. IEEE Computer Press, 1989.

[18] Dimitrovici, C., Hummert, U., and Pétrucci, L. *The properties of algebraic net schemes in some semantics*. L.R.I Rap. de Recherce 539. Univeriste de Paris-Sud, 1990.

[19] Dimitrovici, C. and Hummert, U. *Compositionality of algebraic high-level nets*. Technical report. Institut für Software und Theoretische Informatik, Technische Universität Berlin, 1989.

[20] Dimitrovici, C. and Hummert, U. *Kategorielle konstruktionen für algebraische Petrinetze*. Technical Report 23. Fachbereich Informatik, Technische Universität Berlin, 1989.

[21] Dimitrovici, C. and Hummert, U. *Semantische konstruktionen und kategorien algebraischer netschemata*. Technical Report 12. Fachbereich Informatik, Technische Universität Berlin, 1989.

[22] Ehrig, H. and Mahr, B. *Fundamentals of algebraic specification I: Equations and initial semantics*. Springer-Verlag, Berlin, 1985.

[23] Enderton, H. B. *A mathematical introduction to logic*. Academic Press, New York, 1972.

[24] Engberg, U. and Winskel, G. *Petri nets as models of linear logic*. Technical Report DAIMI PB-301. Computer Science Department, Aarhus University, Denmark, 1990.

[25] Gallier, J. H. *Logic for computer science*. John Wiley & Sons, New York, 1987.

[26] Genrich, H. *Predicate/transition nets.* Petri Nets: Central Models and Their Properties. Lecture Notes in Computer Science 254. Springer Verlag, Berlin, 1986, pp. 207–247.

[27] Genrich, H. and Stankiewicz-Wiechno, E. *A dictionary of some basic notions of net theory.* Net Theory and Applications. Lecture Notes in Computer Science 84. Springer-Verlag, Berlin, 1900, pp. 519–535.

[28] Girard, J.-Y. *Linear logic.* Theoretical Computer Science, 50(1987), pp. 1–102.

[29] Girard, J.-Y. *Quantifiers in linear logic.* Proc. of the SILFS Conference, Cesena, Italy. January 1987.

[30] Girard, J.-Y. *Geometry of interaction I: Interpretation of system F.* Logic Colloquium '88. North-Holland, Amsterdam, 1989.

[31] Girard, J.-Y. *Towards a geometry of interaction.* AMS Conference on Categories in Computer Science 1987. Contemporary Mathematics 92. American Mathematical Society, Providence, Rhode Island, USA, 1989, pp. 69–108.

[32] Girard, J.-Y. *Geometry of interaction II: Deadlock-free algorithms.* COLOG 1988. Lecture Notes in Computer Science 417. Springer-Verlag, Berlin, 1990.

[33] Girard, J.-Y. *La logique linéaire.* Pour La Science, Édition Francaise de Scientific American, 150(1990), pp. 74–85.

[34] Girard, J.-Y. and Lafont, Y. *Linear logic and lazy computation.* TAPSOFT '87. Springer LNCS 250, 1987, pp. 52–66.

[35] Girard, J.-Y. *The system F of variable types, fifteen years later.* Theoretical Computer Science, 45(1986), pp. 159–192.

[36] Girard, J.-Y., Lafont, Y., and Taylor, P. *Proofs and types.* Cambridge University Press, England, 1989.

[37] Gödel, K. *Über eine bisher noch nicht benützte erweiterung des finiten standpunktes.* Dialectica, 12(1958), pp. 280–287.

[38] Goguen, J. A. and Burstall, R. *Institutions: Abstract model theory for specification and programming.* Research Report ECS-LFCS-90-106. Laboratory for Foundations of Computer Science, University of Edinburgh, Scotland, 1990.

[39] Gunter, C. and Gehlot, V. *Nets as tensor theories.* 10th International Conference on Applications and Theory of Petri Nets. Bonn, 1989, pp. 174–191.

[40] Harland, J. and Pym, D. *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, 1990.

[41] Huber, P., Jensen, K., and Shapiro, R. M. *Hierarchies in coloured petri nets.* 10th International Conference on Application and Theory of Petri Nets. Bonn, 1989, pp. 192–208.

[42] Hummert, U. *Algebraische theorie von high-level netzen.* PhD thesis, Technische Universität Berlin, 1989.

[43] Husberg, N. *Petri nets in algebraic theories - a category theory approach.* Technical Report B5. Digital Systems Laboratory, Helsinki University of Technology, Finland, 1988.

[44] Jensen, K. *Coloured Petri nets.* Petri Nets: Central Models and Their Properties. Lecture Notes in Computer Science 254. Springer Verlag, Berlin, 1986, pp. 247–299.

[45] Johnstone, P. T. *Stone spaces.* Cambridge University Press, England, 1982.

[46] Kelly, G. M. and Street, R. *Review of the elements of 2-categories.* Category Seminar. Lecture Notes in Mathematics 420. Springer Verlag, Berlin, 1974. pp. 75–103.

[47] Lafont, Y. *The linear abstract machine.* Theoretical Computer Science, 59(1988), pp. 157–180.

[48] MacLane, S. *Categories for the working mathematician.* Springer Verlag, Berlin, 1971.

[49] Marti-Oliet, N. and Meseguer, J. *From Petri nets to linear logic.* Category Theory and Computer Science. Lecture Notes in Computer Science 389. Springer Verlag, Berlin, 1989, pp. 313–340.

[50] Masseron, M., Tollu, C., and Vauzeilles, J. *Generating plans in linear logic.* Foundations of Software Technology and Theoretical Computer Science. Lecture Notes in Computer Science 472. Springer-Verlag, Berlin, 1990, pp. 63–75.

[51] Meseguer, J. Private Communication, 1991.

[52] Meseguer, J. and Montanari, U. *Petri nets are monoids.* Information and Control, 88(1990), pp. 105–155.

[53] Milner, R. *Communication and concurrency.* Prentice Hall, London, 1989.

[54] Mulvey, C. J. *&.* Rendiconti del Circolo Matematico di Palermo, 12(1986), pp. 99–104.

[55] Niefield, S. B. and Rosenthal, K. I. *Constructing locales from quantales.* Mathematical Proceedings of the Cambridge Philosophical Society, 104(1988), pp. 215–234.

[56] Reisig, W. and Vautherin, J. *An algebraic approach to high level Petri nets.* 8th Workshop on Applications and Theory of Petri Nets. Zaragoza, Spain, 1987, pp. 51–72.

[57] Reisig, W. *Petri nets and algebraic specifications.* Theoretical Computer Science, 80(1991), pp. 1–34.

[58] Scedrov, A. *A brief guide to linear logic.* Bulletin of the European Assoc. for Theoretical Computer Science, 41(1990), pp. 154–165.

[59] Tuominen, H. *Logic in Petri net analysis.* Research Report A5. Digital Systems Laboratory, Helsinki University of Technology, Finland, 1988.

[60] van Glabbeek, R. *Comparative concurrency semantics and refinement of actions.* PhD thesis, Vrije Universiteit te Amsterdam, 1990.

[61] Vautherin, J. *Un modele algebrique, base sur les reseaux de petr, pour l'etude des systemes paralleles.* PhD thesis, Universite de Paris-Sud, 1985.

[62] Vickers, S. *Topology via logic.* Cambridge University Press, England, 1989.

[63] Wadler, P. *Linear types can change the world!* Programming Concepts and Methods. Elsevier Science Publisher B.V. (North-Holland), Amsterdam, 1990. pp. 561–580.

[64] Winskel, G. *Categories of models of concurrency.* Seminar on Concurrency. Lecture Notes in Computer Science 197. Springer Verlag, Berlin, 1984, pp. 246–267.

[65] Winskel, G. *A new definition of morphism on Petri nets.* STACS 84. Lecture Notes in Computer Science 166. Springer-Verlag, Berlin, 1984.

[66] Winskel, G. *Petri nets, algebras and morphisms.* Technical Report 79. Computer Laboratory, University of Cambridge, England, 1985.

[67] Winskel, G. *A category of labelled petri nets and compositional proof systems.* 3rd Symposium on Logic in Computer Science. IEEE Computer Press, 1988, pp. 142–154.

[68] Winskel, G. Private Communication, 1990.

[69] Yetter, D. N. *Quantales and (noncommutative) linear logic.* Journal of Symbolic Logic, 55(1990), pp. 41–64.

A Appendix: Introduction to 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 emphasises mappings before objects. Indeed most of category theory could be written without direct reference to objects.

An other equivalent way to characterise 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. E.g. if we require that $f\,;g = h\,;k$ 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 colimits, which are ways of expressing combinations of objects. Finally we define the notion of adjunction. For missing proofs we refer to [48].

A.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 23
A category $\underline{C}$ is a collection of objects $|\underline{C}|$ such that

- For each pair of objects (a, b) of $\underline{C}$ a set $\mathbf{Mor}_{\underline{C}}(a, b)$ called the *set of morphisms from a to b*, with $\mathbf{Mor}_{\underline{C}}(a, b)$ and $\mathbf{Mor}_{\underline{C}}(a', b')$ 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 $|\underline{C}|$ there is a mapping (composition)

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

  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 $\mathbf{id}_a \in \mathbf{Mor}_{\underline{C}}(a, a)$ which is right identity under ; for the elements of $\mathbf{Mor}_{\underline{C}}(a, b)$ and left identity under ; for the elements $\mathbf{Mor}_{\underline{C}}(b, a)$.
  - ; 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 emphasise 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 2
(The category of sets) The category of sets $\underline{\mathrm{Set}}$ and their mappings. The objects are ordinary sets and the morphisms are ordinary mappings between sets. Composition is the usual composition of mappings. □

Example 3
(Preorder) Let $(E, \preceq)$ be a preordered set. We can view this preordered set as a category $\underline{E}$ as follows: take as objects of $\underline{E}$ the elements of E and for $a, b \in E$ define

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

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

Example 4
(Monoid) A monoid is a set X equipped with a function $\bullet : X \times X \rightarrow X$ (monoid multiplication) and a distinguished element $\mathbf{I}$ (monoid identity) subject to the two laws:

$$\begin{array}{rcl} x \bullet (y \bullet z) & = & (x \bullet y) \bullet z \text{ for all } x, y, z \\ x \bullet \mathbf{I} = & x & = \mathbf{I} \bullet x \text{ for all } x. \end{array}$$

Now a monoid is a category with one object. To see this call the object A, then let $X = \mathbf{Mor}(A, A)$ where $\bullet$ is composition and $\mathbf{I}$ is the identity morphism. □

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 24
The opposite category $\underline{C}^{op}$ is formed by turning around all the arrows in $\underline{C}$. □

The fundamental concept in standard mathematics is the epsilon or $\in$-relation. We will now show how some useful set-theoretical constructions can be reformulated in the language of arrows. The main idea is to transform concepts defined by reference to the "internal" membership structure of a set into an "external" characterization by reference to other sets.

As our first concept we will transform the set-theoretic definition of an injective function into the language of arrows. A function $f : a \rightarrow b$ is said to be injective when for all $x, y \in a$

$$\text{if } f(x) = f(y) \text{ then } x = y \text{ .}$$

To transform this into a statement about arrows we introduce two "parallel" arrows $g, h : c \rightrightarrows a$. If we compose g and h with f we get the following diagram:

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

which commutes if $g\,;f = h\,;f$. Now for an $x \in c$ we have $g(x)\,;f = h(x)\,;f$, or $f(g(x)) = f(h(x))$. But as f is injective this means that $g(x) = h(x)$. Thus we have for an injective f

$$\text{whenever } \; g\,;f = h\,;f \; \text{ then } \; g = h \ .$$

We can naturally dualise the above discussion, in which case we get a characterization of a surjective function.

Now we drop the requirement that f be a function and define monics and epics as follows:

Definition 25
Let $\underline{\mathrm{C}}$ be a category and $g, h : c \to a$ $\underline{\mathrm{C}}$-morphisms. A $\underline{\mathrm{C}}$-morphism $f : a \to b$ is:

$$\text{monic iff } \; g\,;f = h\,;f \quad \Rightarrow \quad g = h \ .$$

□

Definition 26
Let $\underline{\mathrm{C}}$ be a category and $g, h : b \to c$ $\underline{\mathrm{C}}$-morphisms. A $\underline{\mathrm{C}}$-morphism $f : a \to b$ is:

$$\text{epic iff } \; f\,;g = f\,;h \quad \Rightarrow \quad g = h \ .$$

□

Another important morphism type is an isomorphism:

Definition 27
Let $\underline{\mathrm{C}}$ be a category. Then a morphism $f : a \to b$ is an isomorphism iff there exists a morphism

$$k : b \to a \; \text{ such that } \; f\,;k = \mathbf{id}_a \; \text{ and } \; k\,;f = \mathbf{id}_b \ .$$

□

Example 5
In $\underline{\text{Set}}$ monic morphisms are injections and epics are surjections. $\square$

As we are interested in studying structure preserving mappings between objects it seems natural to define mappings between categories. These are called functors.

Definition 28
(Functor) A (covariant) functor from a category $\underline{\text{A}}$ to a category $\underline{\text{B}}$ is a pair of mappings that assign to every object $a \in |\underline{\text{A}}|$ an object $F(a) \in |\underline{\text{B}}|$ and to every morphism $f : a \to b \in \mathbf{Mor}_{\underline{\text{A}}}$ a morphism $F(f) : F(a) \to F(b) \in \mathbf{Mor}_{\underline{\text{B}}}$ such that:

- $F(\mathbf{id}_a) = \mathbf{id}_{F(a)}$ for all $a \in |\underline{\text{A}}|$
- if $f\,;g$ is defined in $\underline{\text{A}}$ then $F(f)\,;F(g)$ is defined in $\underline{\text{B}}$ and $F(f)\,;F(g) = F(f\,;g)$.

We use the notation $F : \underline{\text{A}} \to \underline{\text{B}}$ for a functor F from $\underline{\text{A}}$ to $\underline{\text{B}}$. A contravariant functor $F : \underline{\text{A}} \to \underline{\text{B}}$ is a covariant functor $F : \underline{\text{A}} \to \underline{B}^{op}$. $\square$

Thus a functor is a mapping that respects compositions and identities. Two special kinds of functors deserve mentioning. A bifunctor is a functor $F : \underline{\text{A}} \times \underline{\text{B}} \to \underline{\text{C}}$. An endofunctor is a functor $F : \underline{\text{A}} \to \underline{\text{A}}$.

Example 6
(Monoid homomorphisms) Let $\underline{\text{M}}$ and $\underline{\text{N}}$ be monoids viewed as categories. Then a monoid homomorphism $f : (M, \bullet_M, 0_M) \to (N, \bullet_N, 0_N)$ is a functor $H : \underline{\text{M}} \to \underline{\text{N}}$. $\square$

Example 7
(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. $\square$

There is one functor that we will use in the discussion of the adjunction. It is the functor that maps the morphisms of a category into sets in $\underline{\text{Set}}$. Intuitively it takes two objects a and b and returns the set of morphisms from a to b. This is one of the few times when we have to take into account the possibility of foundational problems, because we have to make sure our morphism sets are "proper" sets. This is an assumption that will be true henceforth and will not be explicitely mentioned.

Proposition 14
Let $\underline{\mathrm{C}}$ be a category. Then there is a functor

$$\mathrm{hom}_{\underline{\mathrm{C}}} : \underline{C}^{op} \times \underline{\mathrm{C}} \to \underline{\mathrm{Set}}$$

where $\mathrm{hom}_{\underline{\mathrm{C}}}(a, b) = \mathbf{Mor}_{\underline{\mathrm{C}}}(a, b)$ is the set of $\underline{\mathrm{C}}$-morphisms from a to b, and

$$\mathrm{hom}_{\underline{\mathrm{C}}}(f, g)(k) = fkg$$

with

$$\begin{array}{rcl} f : a' \to a & \in & \mathbf{Mor}_{\underline{C}^{op}} \\ g : b \to b' & \in & \mathbf{Mor}_{\underline{\mathrm{C}}} \end{array}$$

subject to the commutativity of:

$$\begin{tikzcd} a \arrow[r, "k"] & b \arrow[d, "g"] \\ a' \arrow[u, "f"] \arrow[r, "f\,;k\,;g"'] & b' \end{tikzcd}$$

and

$$\begin{tikzcd} \mathrm{hom}_{\underline{\mathrm{C}}}(a, b) \arrow[d, "\mathrm{hom}_{\underline{\mathrm{C}}}(f,g)"'] \\ \mathrm{hom}_{\underline{\mathrm{C}}}(a', b') \end{tikzcd}$$

Proof: It is easy to check that hom preserves identities and compositions.

□

This proposition allows us now to define:

Definition 29
(hom-functor or representable functor)

The functor $\mathrm{hom}_{\underline{\mathrm{C}}} : \underline{C}^{op} \times \underline{\mathrm{C}} \to \underline{\mathrm{Set}}$ is called the hom-functor for the category $\underline{\mathrm{C}}$. Usually $\underline{\mathrm{C}}$ is clear from the context and we drop the subscript. By fixing a specific object a we can define two functors:

- a covariant hom-functor

$$\mathrm{hom}(a, \_\_) : \underline{\mathrm{C}} \to \underline{\mathrm{Set}}$$

- and a contravariant hom-functor

$$\hom(\_\_, a) : \underline{C}^{op} \to \underline{\mathrm{Set}} \,. \tag{1}$$

□

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 30
(Natural transformation) If $F, G : \underline{\mathrm{A}} \to \underline{\mathrm{B}}$ are functors then a natural transformation from F to G is a rule that assigns to each object $a \in |\underline{\mathrm{A}}|$ a morphism $\eta_a : F(a) \to G(a) \in \mathbf{Mor}_{\underline{\mathrm{B}}}$ in such a way that associated with every morphism $f : a \to b \in \mathbf{Mor}_{\underline{\mathrm{A}}}$ 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 8
A good example of a natural transformation is the evaluation of a function at an argument, (we will return to this example from a different point of view in the next section). 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 $F : \underline{\mathrm{Set}} \to \underline{\mathrm{Set}}$. So for this specific A $eval : F \to I_{\underline{\mathrm{Set}}}$. 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 nauseaum". 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.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 categorial 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 *colimits*. Below when we use the word limit, we usually mean both limits *and* colimits. Our first example of a limit is the categorial product, a generalization of a cartesian product of two sets.

Definition 31
(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 $\underline{C}$ and every pair of morphisms (f, g) 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 32
(Coproduct) Let a, b be objects in $\underline{C}$. A coproduct of a and b in $\underline{C}$ is an object $a + b$ with two morhpisms $\iota_1 : a \to a + b, \iota_2 : b \to a + b$, 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 9
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. □

Products were historically the first example of categorial limits. The characterization of a cartesian product by an universal arrow then helped consolidate the concept.

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 univeral arrow. Formally:

Definition 33
(Initial object) An object $\mathbf{i}$ 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 $\mathbf{i}$ to x. □

Again dually we can ask what an empty product is and end up with the following:

Definition 34
(Terminal object) An object $\mathbf{t}$ in $\underline{\mathrm{C}}$ is said to be a terminal object if for every other object x in $\underline{\mathrm{C}}$ there is only one arrow from x to $\mathbf{t}$. □

Example 10
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. □

It is an easy exercise to prove that all initial (terminal) objects in a category are isomorphic.

This can be generalised to all limits:

Proposition 15
All limits of the same type are isomorphic.

□

Corollary 1

$$\begin{aligned} a \times b &\simeq b \times a, \\ a + b &\simeq b + a \;. \end{aligned}$$

□

A third kind of limit that we will need is called an equaliser. It is a generalization of the following set-theoretical construction.

For a parallel pair of functions $f, g : A \rightrightarrows B$ in <u>Set</u> let E be the subset of A on which f and g agree, i.e.:

$$E = \{x | x \in A \text{ and } f(x) = g(x)\} \;.$$

The inclusion function $i : E \hookrightarrow A$ is called the *equaliser* of f and g. The intuitive meaning of the name is that i "equalises" f and g, i.e $i\,;f = i\,;g$.

By rewriting the above discussion into category theory we get the following two definitions:

Definition 35
Given in <u>C</u> a pair of arrows $f, g : a \to b$ with the same domain a and codomain b, an equaliser 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:

$$\begin{tikzcd}
e \arrow[r, "u"] & a \arrow[r, shift left, "f"] \arrow[r, shift right, "g"'] & b \\
c \arrow[u, "h'"] \arrow[ur, "h"'] & &
\end{tikzcd}$$

□

Definition 36
Given in $\underline{C}$ a pair of arrows $f, g : a \to b$ with the same domain a and codomain b, a coequaliser 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 coequaliser corresponds to an equivalence relation. In set-theoretical terms the coequaliser identifies those elements of b that are the image of the same x of a under the functions f and g.

An other important type of limit is the pullback and its dual the pushout. As this work contains applications of the pushout we will only define it and leave the definition the pullback to the interested reader.

Definition 37
Given in $\underline{C}$ a pair of arrows $f : a \to b$ and $g : a \to c$ with the same domain a, a pushout is given by an object d and arrows $\iota_1 : b \to d, \iota_2 : c \to d$ such that:

- $f\,; \iota_1 = g\,; \iota_2$, and
- given any other object d' and maps $k : b \to d', l : c \to d'$ there exists an unique map $h : d \to d'$ such that the following diagram commutes:

$$\begin{tikzcd} a \arrow[r, "f"] \arrow[d, "g"'] & b \arrow[d, "\iota_1"] \arrow[ddr, "k"] & \\ c \arrow[r, "\iota_2"] \arrow[drr, "l"'] & d \arrow[dr, "h"] & \\ & & d' \end{tikzcd}$$

□

Intuitively what a pushout does is that it patches together the objects c and d at the points specified by the maps g and f. Thus when talking about nets the pushout will consist of the "gluing" together of nets to compound nets.

The above limits and colimits are instances of a general limit construction and colimit construction respectively. We shall not delve into this because the limit and colimits that are interesting from our point of view have been presented above.

A.3 Adjunctions

One could boldly state that an adjunction is the most fundamental concept in category theory. 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 38
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 $f(a) \preceq_B b \Leftrightarrow a \preceq_A g(b)$ 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 65), 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 generalise the notion of a Galois connection to an adjunction as follows:

Definition 39
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 classical example of an adjunction can be found in a Cartesian Closed Category (CCC).

Definition 40
A category $\underline{C}$ is cartesian closed iff

- It has all finite products
- Fix an arbitrary $c \in |\underline{C}|$. Every functor $F : \underline{C} \to \underline{C} = b \mapsto b \times c$ has a right adjoint $G : \underline{C} \to \underline{C} = a \mapsto \mathbf{Mor}_{\underline{C}}(a, c)$ such that the diagram below commutes:

$$\begin{tikzcd} c^b \times b \arrow[r, "\varepsilon^{b,c}"] & c \\ a \times b \arrow[u, "f^* \times \mathbf{id}_b"] \arrow[ur, "f"'] & \end{tikzcd}$$

□

The two adjoints define the adjunction:

$$(F, G, \eta, \varepsilon) : \underline{C} \to \underline{C}$$

where:

$$\begin{aligned} b\eta &= \mathbf{Mor}_{\underline{C}}(b, \mathbf{Mor}_{\underline{C}}(c, b \times c)), \\ a\varepsilon &= \mathbf{Mor}_{\underline{C}}(\mathbf{Mor}_{\underline{C}}(c, a) \times c, a)\ . \end{aligned}$$

and F, G are as defined before.

Cartesian closed categories also suggest an equivalent definition of an adjunction as an isomorphism between hom-functors:

Proposition 16
An adjunction $(F, G, \eta, \varepsilon)$ between locally small categories $\underline{A}$, $\underline{B}$ gives rise to and is determined by a natural isomorphism $\hom_{\underline{A}}(F(a), b) \simeq \hom_{\underline{B}}(a, G(b))$ for all $a \in |\underline{A}|, b \in |\underline{B}|$. □

This also gives us a second (maybe more natural) definition of a cartesian closed category:

Definition 41
A category $\underline{\mathrm{C}}$ is cartesian closed iff:

- It has all finite products
- For all objects $a, b, c \in |\underline{\mathrm{C}}|$ there exists an object c^b called the "power object of b w.r.t. c" such that

$$\hom_{\underline{\mathrm{C}}}(a \times b, c) \simeq \hom_{\underline{\mathrm{C}}}(a, c^b) \ .$$

□

The isomorphism is the categorial equivalent to the operation of "currying" where a function $f : a \times b \to c$ is viewed as a *higher order* function $\Lambda(f) : a \to (b \to c)$ taking one argument and returning a function taking the other argument.

The most important property of an adjunction is expressed by the following fact.

Theorem 10
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 colimits and G preserves limits. □