Towards Linear Logic Petri Nets

From P/T-Nets to Object Systems

Berndt Farwer

Universität Hamburg
Fachbereich Informatik
Vogt-Kölln-Straße 30
22527 Hamburg
Germany

farwer@informatik.uni-hamburg.de

Abstract

Linear Logic [Gir87] has been shown to incorporate a fragment suitable for representing P/T-nets and giving an interleaving semantics to the computations of such nets (e.g. [Bro89], [MOM89], [EW90]). This result is generalized to coloured nets. Furthermore a new kind of high-level nets is defined: Linear Logic Petri Nets (LLPN). These nets are used as an intuitive semantics to well-known and new high-level net concepts like object systems ([Val96b]) and agent orientation. Related topics have been addressed in [Far96] and [Far98].

keywords: Linear Logic, Petri nets, object systems, concurrency

Zusammenfassung

Die Lineare Logik [Gir87] enthält ein Fragment, das sehr gut zur Darstellung von S/T-Netzen geeignet ist. In [Bro89], [MOM89] und [EW90] wird gezeigt, daß dieses Fragment der Linearen Logik Petrinetzen eine (interleaving) Semantik gibt. Diese Resultate werden auf gefärbte Petrinetze erweitert. Außerdem wird eine neue Klasse von High-Level-Netzen eingeführt: Linear Logic Petri Nets (LLPN). LLPNs bieten die Möglichkeit, höheren Netzkonzepten als intuitive Semantik zu dienen. In diesem Zusammenhang werden insbesondere Objektsysteme ([Val96b]) und Agentenorientierung betrachtet. Diese Arbeit stellt eine Fortsetzung der Arbeiten [Far96] und [Far98] dar.

Schlagwörter: Lineare Logik, Petrinetze, Objektsysteme, Nebenläufigkeit

Contents

Chapter 1

Introduction

Petri nets have been used successfully throughout the last three decades for the specification and analysis of a diverse range of problems. The theory of Petri nets has had direct impact on computing and information sciences but the scope in which applications of the theory can be found is not limited to these. In fact there are many other disciplines that are influenced by Petri net theory in some way or another.

The design and analysis of parallel and distributed algorithms has benefited a lot from the research in Petri nets. The basic model of place/transition nets is well-studied, so that there is a wealth of results and techniques available. More recent development has shown that this basic model—even though it is well-understood—somewhat lacks in one aspect essential for the design of complex systems leading to some conservative extensions of the basic model. Further high-level concepts have been added since, so that we are faced by an overwhelming diversity of net concepts.

The foremost problem when extending some net formalism is the (re)definition of the occurrence rule. This should be done with extreme care, not only to avoid undesired behaviour in the nets of the newly defined class, but also to make sure that as many results from the standard theory are preserved for the new class.

In this paper we define a class of Petri nets that have Linear Logic formulae as tokens. These nets—called Linear Logic Petri nets or LLPNs—are capable of capturing the behaviour of various other high-level nets and can be seen as a step towards a uniform semantics for concurrency models. LLPNs are defined on purely logical grounds.

1.1 Some Notes on Notation

We will use varying notions for multisets throughout this text according to the situation. It is sometimes more convenient to view a multiset as a mapping from the underlying domain to the natural numbers, giving the multiplicity for each element explicitly. On other occasions it is easier to use a set-like notation in which we make no notational difference compared with usual sets, for example $\mathbf{m} := \{1, 1, 4\}$ denotes the multiset over $\mathbb{N}$ such that $\mathbf{m}(1) = 2$ and $\mathbf{m}(4) = 1$, whereas for all $x \in \mathbb{N} \setminus \{1, 4\}$ the value of $\mathbf{m}(x)$ is 0. We believe that the meaning will be apparent from the context.

For multisets we will usually use formal sums and represent the union of two multisets $\mathbf{m}$ and $\mathbf{m}'$ by $\mathbf{m} + \mathbf{m}'$. By $\mathbf{m} - \mathbf{m}'$ we denote the multiset difference of $\mathbf{m}$ and $\mathbf{m}'$.

Capital Greek letters are used to denote environment variables that can be instantiated with multisets of formulae. Variables that may be bound to a formula are usually taken from the rear of the alphabet.

In logic formulae it is generally assumed that conjunctions and disjunctions bind stronger than implications. Quantifyers and negation are assumed to bind stronger than other connectives.

For a Petri net transition t the preset and postset are denoted ${}^\bullet t$ and $t^\bullet$, respectively.

Chapter 2

Basics

This chapter briefly summarizes some results concerning the relationship between Linear Logic and Petri nets.

2.1 Linear Logic and P/T-nets

Linear Logic has been shown to be well suited for describing Petri nets and their dynamics, few attempts have been made though to sophisticate linear logic for analysing nets and proving properties further than the reachability of certain markings[1]. In fact the main value of Linear Logic for Petri nets is in giving them a purely logical semantics.

Before going into the details of Petri net representation we will briefly and informally characterise the connectives of linear logic, the main idea of which is to split the conjunction (resp. disjunction) into two different versions, one that is resource sensitive and another that behaves more or less like the classical connective. In order to maintain the power of classical calculi one has to introduce modalities, the so called exponentials.[2]

The linear constants and connectives used in the context of Petri nets are:

Mutiplicatives:

- $\otimes$ (*times, tensor*) is the multiplicative resource sensitive version of classical conjunction, e.g. $A \otimes B$ in linear logic means that both resources A and B are present at the same time, whereas $A \otimes A$ means that two instances of the same resource are present.
- $\parr$ (*par*) is the multiplicative disjunction, $A \parr B$ can be interpreted as: if not A then B.

[1] Some attempts have been made to extend Linear Logic by some concepts from temporal logics in [Tan97] and [KI97].

[2] Actually Linear Logic arises when thinking about the so-called structural rules of a Gentzen-like sequent calculus of intuitionistic logic. The removal of the contraction rule and the weakening rule leads to the multiplicative fragment of Linear Logic while their re-introduction as logical rules in the context of the exponential modalities regains the power of classical and intuitionistic calculi.

- $\multimap$ (*entails*) is the multiplicative implication, such that $A \multimap B$ means that we get one new instance of resource B while consuming exactly one instance of resource A.
- The multiplicative constants are 1 and $\bot$, where 1 is the unit of the multiplicative conjunction, meaning truth only in isolation, and $\bot$ is the unit of *par* representing a placeholder for nothingness.

Additives:

- The additive conjunction & (*with*) expresses a kind of deterministic choice, e.g. in a situation where both resources are offered to you but you cannot grab both of them, $A\&B$ is the representation of your choice between A and B.
- The additive disjunction $\oplus$ (*plus*) on the other hand represents nondeterminism or a choice on the systems side, i.e. given $A \oplus B$ it is at the system resource managers discretion to give you either A or B. You can only be sure not to leave empty handed.
- The units of the additive conjunction and disjunction are $\top$ and 0, representing true truth and falsity, respectively (i.e. truth or falsity in any context).

Negation: Linear *negation* $(\cdot)^\bot$ could be called a dept in monetary terms. In general $A^\bot$ is an input slot for using up one instance of resource A.

Exponentials:

- The exponential ! (*of course*) is the storage operator, also called the operator of reusability, which makes a resource arbitrarily available.
- The second exponential ? has no immediate meaning in the context of Petri nets. It can be thought of as an operator of debts that is dual to !.

2.2 Basic Relationship

We will give a short survey of the work done by C. Brown, N. Martí-Oliet and J. Meseguer, U. Engberg and G. Winskel. We use the notion of a marked net, i.e. that of an instantaneous description of a net. Only formulae from propositional Linear Logic that contain the tensor product $\otimes$, linear implication $\multimap$, the additive connective of choice &, the storage-operator ! and the constant 1 will be used in this section.

2.3 Terminology

The Linear sequent calculus will be used in the following expositions of results. We denote sequents by $\Gamma \vdash \Delta$, where Γ and Δ are multisets of formulae, and $\vdash$ is a metasymbol that has the meaning of entailment in the calculus. The multisets are usually written in list notation, omitting any superfluous braces or parentheses, i.e. a sequent will often be written as $A_1, \ldots, A_n \vdash B_1, \ldots, B_m$. The semantics given to such sequent in the classical

sequent calculus introduced by Gentzen in [Gen35] is $A_1 \wedge \cdots \wedge A_n \vdash B_1 \vee \cdots \vee B_m$. For linear logic the multiplicative fragment is used to give the semantics to such sequent, i.e. $A_1 \otimes \cdots \otimes A_n \vdash B_1 \parr \cdots \parr B_m$ which is equivalent to the occasionally used one sided sequent $\vdash A_1^{\perp} \parr \cdots \parr A_n^{\perp} \parr B_1 \parr \cdots \parr B_m$ or $\vdash A_1^{\perp}, \ldots, A_n^{\perp}, B_1, \ldots, B_m$ for short.

2.4 Sequent Calculus for Linear Logic

$$A \vdash A (\text{Identity}) \qquad \vdash 1 (1)$$

$$\frac{\Gamma \vdash A \quad \Delta, A \vdash B}{\Gamma, \Delta \vdash B} (\text{Cut}) \qquad \frac{\Gamma, A, B, \Delta \vdash C}{\Gamma, B, A, \Delta \vdash C} (\text{Exchange})$$

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

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \& B} (\& \text{R}) \qquad \frac{\Gamma, A \vdash C}{\Gamma, A \& B \vdash C} (\& \text{L1}) \qquad \frac{\Gamma, B \vdash C}{\Gamma, A \& B \vdash C} (\& \text{L2})$$

$$\frac{\Gamma \vdash A \quad \Delta, B \vdash C}{\Delta, \Gamma, A \multimap B \vdash C} (\multimap \text{L})$$

$$\frac{\Gamma, !A, !A \vdash B}{\Gamma, !A \vdash B} (\text{Contraction}) \qquad \frac{\Gamma, A \vdash B}{\Gamma, !A \vdash B} (\text{Dereliction})$$

$$\frac{\Gamma \vdash B}{\Gamma, !A \vdash B} (\text{Weakening})$$

Table 2.1: Inference Rules for $\mathcal{L}_{Petri}$

Only the inference rules from the fragment of linear logic shown in table 2.1 are needed to show the derivability of the canonical Linear Logic formula for a given nets system coincides with the reachability in P/T-nets. We use the two-sided version of the sequent calculus rules here, as it appeals more naturally to our intuition.

Remark. Note that although only intuitionistic rules are considered above, which allow but for interleaving semantics, the full classical Linear Logic calculus is suitable for real concurrency semantics.

Let us take a look at an example of a simple Petri net from which we will derive a Linear Logic representation. In the remainder of this section we will use the following abbreviation:

$$a^n := \underbrace{a \otimes \cdots \otimes a}_{n}$$

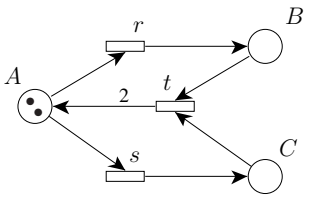

Figure 2.1: A Simple Petri net from the Example on page 6

Example. Starting from the Petri net system shown in Figure 2.1 we can obtain a natural set of formulae from the fragment $\mathcal{L}_{Petri}$ of Linear Logic to describe it. We start by describing the current marking of the net by introducing one resource of type p_i for each token on place p_i and writing them as a tensor product. In our net the current marking is therefore A^2 indicating the presence of two tokens on place A.

We shall now construct subformulae for each transition of the net:

- Firing t is represented by the formula $B \otimes C \multimap A^2$, meaning that one token each is consumed from places B and C, while two tokens are produced on place A.

- For transitions r and s we have: $A \multimap B$ and $A \multimap C$, respectively.

As usual every transition may fire as often as desired, as long as the preconditions are satisfied. Therefore we have to precede each formula that represents a transition by the storage operator *of course* (!). Hence the complete description of transition t is: $!(B \otimes C \multimap A)$, making t available *ad infinitum*. By putting all subformulae together we arrive at an instantaneous description of the net system:

$$A^2 \otimes !(A \multimap B) \otimes !(A \multimap C) \otimes !(B \otimes C \multimap A^2)$$

In the manner outlined in the previous example it is possible to construct for every net system $\mathcal{S}(\mathbf{m}) = \langle \mathcal{N}, \mathbf{m} \rangle = \langle P, T, F, W, \mathbf{m} \rangle$, its canonical formula $\Psi_{\mathcal{S}(\mathbf{m})}$, by forming the tensor product of the following formulae:

- For a transition t with non-empty preconditions ${}^\bullet t$ and non-empty postconditions $t^\bullet$, construct

$$! \left(\bigotimes_{p \in {}^\bullet t} p^{W(p,t)} \multimap \bigotimes_{q \in t^\bullet} q^{W(t,q)} \right),$$

- In the special cases where a transition t has no preconditions (i.e. a *source transition*) construct for each such transition the formula

$$! \left(1 \multimap \bigotimes_{q \in t^\bullet} q^{W(t,q)} \right) \text{ or equivalently } ! \left(\bigotimes_{q \in t^\bullet} q^{W(t,q)} \right),$$

- For all transitions t without any postconditions (i.e. *sink transitions*) construct the linear logic formula

$$! \left(\bigotimes_{p \in {}^\bullet t} p^{W(p,t)} \multimap \bot \right) \text{ or equivalently } ! \left((\bigotimes_{p \in {}^\bullet t} p^{W(p,t)})^\bot \right),$$

- Construct for the current marking $\mathbf{m}$ and all places $p \in P$ with $\mathbf{m}[p] = n$, $n \geq 1$ the formulae p^n. Thus for the complete marking

$$\bigotimes_{\substack{p \in P \\ \mathbf{m}[p] \geq 1}} p^{\mathbf{m}[p]}$$

Having now a representation of a Petri net in Linear Logic we can prove the following soundness and completeness theorem.

Theorem 2.4.1 *A marking $\mathbf{m}'$ is reachable in $\mathcal{S}(\mathbf{m})$ iff for the corresponding canonical formulae the sequent $\Psi_{\mathcal{S}(\mathbf{m})} \vdash \Psi_{\mathcal{S}(\mathbf{m}')}$ is provable in $\mathcal{L}_{Petri}$.*

Proof. See [Bro89] for a detailed proof using induction on the number of steps made in the derivation, especially looking at the last rule used for the *only if*-branch. The *if* branch is straightforward. □

Chapter 3

Linear Logic and Coloured Nets

In coloured Petri nets the marking of one place is described by a multiset of typed (or coloured) tokens. Hence the marking of an entire net requires a set of multisets each qualified by the name of the respective place (if the places have pairwise disjoint names).

One advantage of coloured Petri net is that the resulting net is smaller and subnets of similar structure can easily be reused. This is simply done by folding in two subnets of a P/T-net and adding type (colour) restrictions to certain places. This also means that for any cpn there is a way to unfold it to derive an equivalent P/T-net for which there exists a formula in Linear Logic appropriately specifying the nets behaviour.

More formally a coloured Petri net is a quintuple $P = (P, T, C, W, m_0)$, where P is a set of places and T is a disjoint set of transitions. Furthermore C is the colour function defined from $P \cup T$ into non-empty sets, W is the weight or incidence function, and m_0 is the initial marking of the net .

According to Jensen [Jen92] a coloured Petri net can be transformed into a place/transition-net by replacing each place p with a set of places $C(p)$ (one for each kind of tokens that p may hold) and replacing each transition t with a set of transitions $C(t)$ (one for each way in which t may fire).The relationship between the new places and transitions are determined by the corresponding elements in the matrix determined by the function $W(p, t)$.

There is an obvious way in which coloured Petri nets are representable within the same fragment of Linear Logic used in the preceding sections. The encoding used here is easily arrived at by a standard unfolding of a coloured net. One problem arises when considering infinite colour domains: In the unfolding there may be infinitely many transitions for each transition of the coloured net that has an incoming or outgoing arc labelled by a variable of an infinite colour domain. In this case we would have to use infinitary Linear Logic formulae as considered in [Far96]. If we restrict ourselves to finite colour domains the canonical Linear Logic formula is constructed as in the following example.

Example. Consider the coloured transition depicted below and assume the multiset marking $\mathbf{m}[A] = \{1, 1, 4\}, \mathbf{m}[B] = \{4\}, \mathbf{m}[C] = \{\}, \mathbf{m}[D] = \{\}$

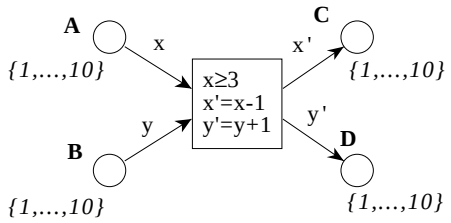

Figure 3.1: Example of a coloured transition

This excerpt from a coloured Petri net can be represented by

$$A_1 \otimes A_1 \otimes A_4 \otimes B_4 \otimes \bigotimes_{\substack{x\in\{1,\ldots,10\}\\ y\in\{1,\ldots,10\}\\ x'\in\{1,\ldots,10\}\\ y'\in\{1,\ldots,10\}\\ (x\geq 3)\wedge(x'=x-1)\wedge(y'=y+1)}} !(A_x \otimes B_y \multimap C_{x-1} \otimes D_{y+1})$$

The following slightly more complex example is taken from [Val98a]. It shows the coding of a complete coloured net.

Example. The coloured net in figure 3.2 describes the starting situation for a car race. The colour sets used for the places are: $cars = \{a, b\}, starter = \{s\}, ready = \{rsa, rsb\}$. The arc inscriptions bear multisets of variables (x, y, z) or constants (s, rsa, rsb, ssa, ssb).

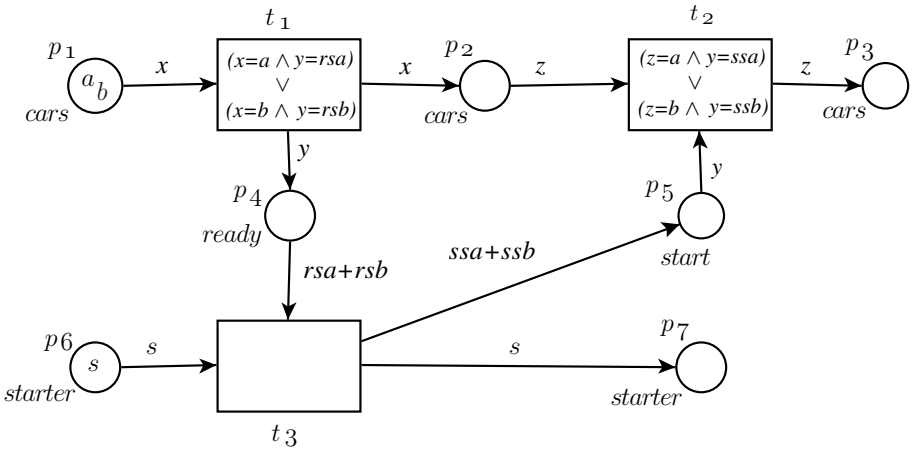

Figure 3.2: Starting a car race as a CPN

A Linear Logic representation of this net is given by :

initial marking:	$p_{1_a} \otimes p_{1_b} \otimes p_{6_s} \otimes$
first mode of t_1:	$\otimes!(p_{1_a} \multimap p_{2_a} \otimes p_{4_{rsa}})$
second mode of t_1:	$\otimes!(p_{1_b} \multimap p_{2_b} \otimes p_{4_{rsb}})$
first mode of t_2:	$\otimes!(p_{2_a} \otimes p_{5_{ssa}} \multimap p_{3_a})$
second mode of t_2:	$\otimes!(p_{2_b} \otimes p_{5_{ssb}} \multimap p_{3_b})$
coding of t_3:	$\otimes!(p_{6_s} \otimes p_{4_{rsa}} \otimes p_{4_{rsb}} \multimap p_{7_s} \otimes p_{5_{ssa}} \otimes p_{5_{ssb}})$

It is easily seen from the first example above that the guard expressions are not incorporated directly in the logic, but are used as set theoretic expressions in the construction. The expansion of such formulae is always finite if the colour sets are finite. Otherwise they yield infinitary formulae.

Chapter 4

Combining Petri Nets and Linear Logic

In the following sections we use Linear Logic formulae not only as inscriptions of Petri Nets—i.e. as arc inscriptions and guards—but more importantly as tokens that move through a system represented by the Petri net. The evolution of such tokens is determined by the rules of the used logic and by the occurrence rule of the net. We first give some idea of what behaviour we would like the token formulae to display and then give the formal definition of a LLPN in section 4.1. An informal discussion of various possibilities for the occurrence rule follows in section 4.2 which leads to a more formal account of the version preferred by he author in 4.2.1 . The last section of this chapter is devoted to some examples for the use of LLPNs.

4.1 Linear Logic Petri Nets

What follows is the basic definition of a Linear Logic Petri net. In this most general version we do not have any restrictions on the fragment of Linear Logic used in the inscriptions or in the tokens themselves. It turns out, though, that it is useful to limit the possible behaviour in practice, when studying this class of nets as a possible semantics for different high-level net concepts.

Definition 4.1.1 (Linear Logic Petri Net (LLPN)) *A Linear Logic Petri net* $LLPN = \langle \mathcal{N}, \Lambda, g, W, \mathbf{m}_0 \rangle$ *consists of a net* $\mathcal{N} = \langle P_\mathcal{N}, T_\mathcal{N}, F_\mathcal{N} \rangle$*, where as usual we assume the sets of places and transitions are finite of the form* $P_\mathcal{N} = \{p_{\mathcal{N},1}, \ldots, p_{\mathcal{N},n_\mathcal{N}}\}$ *and* $T_\mathcal{N} = \{t_{\mathcal{N},1}, \ldots, t_{\mathcal{N},m_\mathcal{N}}\}$.

The logic-colour-function Λ *is a mapping from* $S \cup T$ *into fragments of Linear Logic.* $L_{\Lambda(p)}$ *is used to denote the set of terms that may appear as the colour of a token for place* p*. The Linear Logic sequents that may be used for the guard function* $g : T \longrightarrow pow(L^s_{\Lambda(t)})$ *of a transition* $t \in T_\mathcal{N}$ *is determined by* $L^s_{\Lambda(t)}$ *the set of sequents over* $\Lambda(t)$.

W *is a mapping with domain* $(P \times T) \cup (T \times P)$*, such that if* $\{x, y\} = \{p, t\}$*, then* $W(x, y)$ *is a multiset of variables from* $L_{\Lambda(p)}$*. Moreover, for* $(x, y) \notin F$ *we let* $W(x, y)$ *be*

the empty multiset.

A marking of LLPN is a mapping $\mathbf{m}$ *with domain* P*, such that* $\mathbf{m}(p)$ *is a multiset over* $L_{\Lambda(p)}$*.* $\mathbf{m}_0$ *is the initial marking of LLPN.*

4.2 An Occurrence Rule for LLPNs

This section aims at giving some hints as to what the occurrence rule should incorporate to make LLPNs a useful tool. But LLPNs must not be understood as a modelling tool, rather we suggest to view LLPNs as a formal but easily comprehendible semantics in a uniform setting for other modelling formalisms, such as high-level nets. This leads to some requirements that are incorporated in the approach of LLPNs and the corresponding occurrence rule. The basic concept of a (Linear) Logic Petri Net consists of the following ideas:

- use logical formulae as tokens
- token formulae may evolve within a place according to the derivation rules of the logic calculus. No transition has to be fired for this kind of dynamic behaviour. [1]
- token formulae are temporarily bound to variables that constitute the arc inscriptions while a transition is executed.
- transitions are guarded by logic formulae—actually Gentzen-style sequents—including the variables from their incoming and outgoing arcs. If there exists such a binding of token formulae to arc variables that satisfies all guard formulae of the transition in question, we say that the transition is enabled in this particular binding. The marking of the net is updated according to the enabling binding in case the transition decides to fire.

In some cases it may be necessary to keep a limited record of the history of the net execution. This becomes evident, for instance, when trying to model the exact behaviour of object systems from Valk's original definition. A marking of a LLPN can then optionally be viewed as a pair (ϕ, ψ) where ϕ is a mapping from places of $\mathcal{N}$ to multisets of token formulae and ψ is map from places to multisets of interaction tags[2].

[1] One may object that this design decision undermines a basic rule of Petri nets, namely that no action can occur unless some transition fires. We have nevertheless chosen to break this rule, as there is an easy way out: Think of an invisible transition attached to each place of the underlying net in the manner of a simple loop. Let the guard of these transitions enable it if and only if there is a token formula in the place and let the transition replace it with a derived formula according to the logic calculus.

[2] Interaction tags are used to keep track of which synchronized transitions are fired in each copy of the token formula that concurrently moves through the net. They can be used to check whether two copies of a formula can be joined by a join transition in the sense of object systems, i.e. iff they represent an object net that has evolved in two different ways but can be described by two subprocesses that are consistently extendible to a common object net process.

Let $M = (\phi, \psi)$ and $M' = (\phi', \psi')$ be markings of a LLPN.
$M \Rightarrow M'$ iff one of the following holds:

1. no interaction, no transition occurrence:

   $\psi(p) = \psi'(p)$ and $\phi(p) \vdash \phi'(p)$

2. transport occurrence of t:

   If there is a binding β satisfying $g(t)(W({}^\bullet t, t) + W(t^\bullet, t))$ and $\forall p \in P \,.\, \phi(p) \geq \beta(p, t)$ then

   $$\phi'(p) = \beta(W(t, p)) + \phi(p) - \beta(W(p, t))$$

   and

   $$\psi(p) = \psi'(p)$$

3. interaction occurrence:

   as above, but the satisfying formula may use some interaction tokens from ψ which are consumed. Thus $\psi' \subset \psi$.

We will not go into any more detail of this special kind of markings. For the remainder of this paper we focus on the simpler model where the interaction tokens are omitted.

4.2.1 Formalizing the Occurrence Rule

Formally a marking of a LLPN is defined as a vector of multisets of logic formulae[3] which leads to the following definition of the occurrence rule for LLPNs. A t-binding is an assignment of formulae to the set of variables used within arc inscriptions[4] of the arcs relevant to transition t. A mapping $\beta : {}^\bullet t \cup t^\bullet \longrightarrow L_\Lambda$ serves as such a t-binding.

Definition 4.2.1 *Let $LLPN = \langle \mathcal{N}, \Lambda, \tau, W, \mathbf{m}_0 \rangle$ be a Linear Logic Petri net and $t \in T_\mathcal{N}$. We say t is* enabled with binding β *iff there exists a t-binding satisfying* g(t) *such that there is one occurrence of $\beta(X)$ in the place $p \in {}^\bullet t$ for each occurrence of X in $W(p, t)$, i.e.*

$$\forall p \in {}^\bullet t \,.\, \forall X \in W(p, t) \,.\, \mathbf{m}(p)(\beta(X)) \geq W(p, t)(X)$$

Instead of saying transition t is enabled with binding β we sometimes use the expression: t is enabled in colour β to stress the resemblance to coloured nets. The satisfaction of a guard formula is defined as its syntactic derivability in the Linear Logic sequent calculus.

An enabled transition should of course be able to fire, changing the marking of the net as follows: The formula occurrences bound to the variable of incoming arcs in a t-binding are removed from the current marking. On the other hand there will be created new instances of the formulae bound to outgoing arcs in their respective target places. Note that we are not presently considering any capacities for the places of a LLPN.

[3] A marking of a LLPN assigns to each place a multiset of formulae.

[4] Each variable is assigned a unique formula even if it has multiple appearances in the multiset union of the arc inscriptions. Therefore it is sufficient for β to be defined on the set of relevant variables. Additional auxiliary variables may be used in the guards.

Definition 4.2.2 *Let $LLPN = \langle \mathcal{N}, \Lambda, \tau, W, \mathbf{m}_0 \rangle$ be a Linear Logic Petri net and let $t \in T_{\mathcal{N}}$ be enabled with binding β. Then t may fire changing the marking $\mathbf{m}$ to reach the* follower marking $\mathbf{m}'$ *according to the following rule for every place p in P and every variable X that appears in the arc inscription of any arc connected to t:*

$$\mathbf{m}'(p)(\beta(X)) = \mathbf{m}(p)(\beta(X)) - W(p,t)(\beta(X)) + W(t,p)(\beta(X))$$

In addition to the changing of a marking by firing a transition it is possible to make standard Linear Logic deductions on the terms that represent the tokens, thus changing the marking within the same place without the need of a transition occurrence. We will call this type of dynamic behaviour an *autonomous deduction step* of the LLNP.

4.3 Some Examples

In this section we give a couple of examples to help get acquainted with the formalism of LLPNs. The first example (figure 4.1) shows a very simple Linear Logic Petri Net consisting of one transition and one place only. Let the transition be guarded by the guard function assigning to the arc variables x and y the sequent $x \vdash y$.

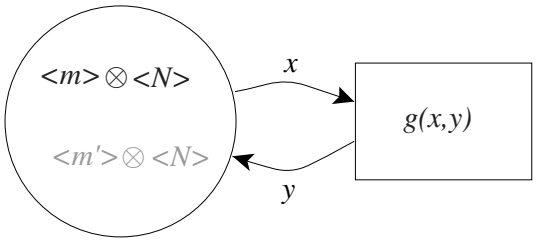

Figure 4.1: LLPN executing a P/T-net

Without restriction of the logic calculus this net allows any logical derivation of formulae residing in the place. But actually the transition and guard are superfluous in this case as the token formulae are allowed to evolve according to the rules of the calculus without using the transition. When restricting the logic to $\mathcal{L}_{Petri}$ we have a representation of all sequential executions of place/transition nets as discussed in chapter 2.

If we allow some kind of occurrence checking in the guard function (see section 4.2.1 we can allow some real concurrency[5] in a restricted way.

Let us turn to a second example in figure 4.2: Assume you want to bake an apple cake. You need a recipe including the baking instructions and all of the ingredients for the dough, you prepare the dough cut three apples into quarters, put them on top of the dough and bake the cake in the preheated oven. Consider the following net that takes a recipe and the ingredients at hand as a token in the place START.

[5]The semantics given to P/T-nets in chapter 2 and thus in the first example is an interleaving semantics.

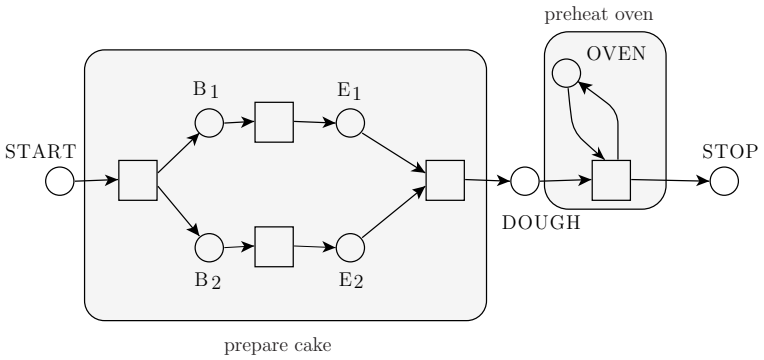

Figure 4.2: Basic structure of processing a recipe.

Let the recipe be given by the Linear Logic formula

$$!((\bigotimes_{j\in J} I_j) \multimap D)\otimes!(A^3 \multimap Q^{12})\otimes$$

$$!(D \otimes Q^{12}\otimes < 1 > \multimap R)\otimes$$

$$!(< 2 > \otimes R \otimes 200 \multimap C \otimes 200)\otimes$$

$$\bigotimes_{j\in J} I_j \quad \otimes A \otimes A \otimes A$$

where the I_j represent the ingredients with a suitable index set J. The ingredients are used to make the dough (D) and the apples (A) are cut into quarters (Q). Having done this the cake is ready (R) to be baked int the preheated oven that produces the cake C. Any unused resources are retained throughout the process of baking the cake.

Taking a closer look at some parts of the net we can see that the preparation of the dough is controlled by four transitions t_1 to t_4 in figure 4.3.

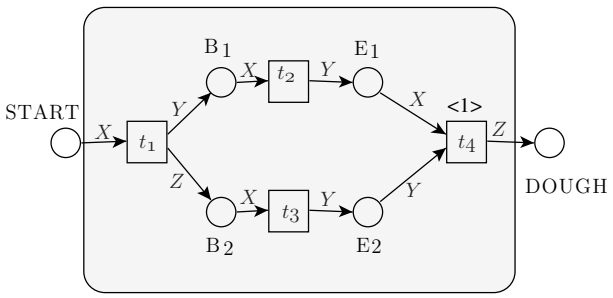

Figure 4.3: Preparing the dough ...

The first transition $t_.1$ takes the recipe and all other resources and divides them into some partition by the guard sequents $X \vdash Y \otimes Z$. This allows for a number of different

partitions, one of which is the partition in which the ingredients for the dough and the accompanying instructions from the recipe are bound to Y and the apples and the rest of the recipe are bound to Z. Now some actions (derivations) may be done concurrently. The transitions t_2 and t_3 do have the guard true such that the only derivations that may happen are those that can be performed on the token formulae by the basic calculus, i.e. autonomous evolution. t_4's guard then simply joins the partitioned formula by the guard $X \otimes Y \vdash Z$. The tag $< 1 >$ is used so that the joining of the subformulae and cake becoming ready to be baked (R) are synchronized.

The oven (figure 4.4) is initialized with a starting temperature of 20 degrees Celsius (room temperature), i.e. let OVEN be marked with 20 representing the initial temperature. The transitions marked $+$ and $-$ are guarded by the sequents

$$\bigotimes_{i \in \{15,\ldots,249\}} !\,(i \multimap i+1) \otimes X \vdash Y \otimes \bigotimes_{i \in \{15,\ldots,249\}} !\,(i \multimap i+1)$$

and

$$\bigotimes_{i \in \{16,\ldots,250\}} !\,(i \multimap i-1) \otimes X \vdash Y \otimes \bigotimes_{i \in \{16,\ldots,250\}} !\,(i \multimap i-1)$$

respectively.

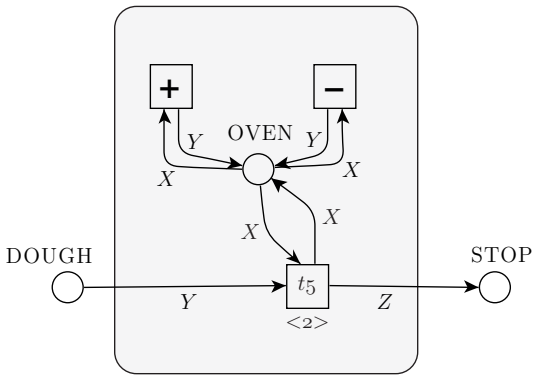

Figure 4.4: ... and baking the cake.

The guard of t_5 must then check the following:

$$Y \vdash R \otimes \Gamma$$

and

$$Z \vdash C \otimes \Gamma$$

and

$$< 2 > \otimes R \otimes \Gamma \otimes \mathrm{X} \vdash \mathrm{Z} \otimes \mathrm{X}$$

Here Γ is the tensor product of the Linear Logic representation of the recipe and all additional resources that were present in the beginning but were not needed.

Of course the net simplifies the task and does not optimally represent it, i.e. there could have been more concurrency[6] involved in baking the cake, but after all this was only meant as an example for the principles of Linear Logic Petri nets.

[6]The partitioning of the task in t_1 does not force a concurrent processing as the recipe and all necessary ingredients may be bound to the same variable and thus be processed sequentially!

Chapter 5

LLPNs and High-Level Nets

We will illustrate the use of LLPNs for giving semantics to some well-known high-level net models. We will focus on the key concepts of each formalism, giving examples of how to represent them in Linear Logic Petri nets. In each case we conclude by giving an example followed by a more general discussion on the specific representation and future work to be done.

5.1 Elementary Object Systems

The main idea of object systems is the use of Petri net systems as tokens in a Petri net. It is an extension of task systems as defined in [Val91]. For this reason a distinction is made between the so-called system net and the object nets. The tokens in object systems may exhibit a dynamic behaviour as they are themselves Petri nets. In addition they can synchronize with one another[1] or with the system net. The occurrence rule for object systems is constructed to allow the distributed parallel execution of an object net in the presence of a strict fork and join structure, i.e. partially executed object nets may only be joined if their processes are compatible in the sense that they can all be extended to a valid object net process.

5.1.1 LLPNs and synchronization

There is a straightforward way to represent synchronization constraints in Linear Logic Petri nets based on the standard representation of transitions in Linear Logic formulae. In the same way the places of the object net are represented as resources, i.e. propositional variables in the logic calculus, we can force a synchronisation by demanding the presence of a resource that can only be made available by the guard of the transition the synchronization shall be enforced with.

Example. Let us take a look at the following excerpt from an object system and construct a Linear Logic Petri net to represent the situation:

[1] Synchronization with other object nets is not in principle restricted to the case where they occupy the same place.

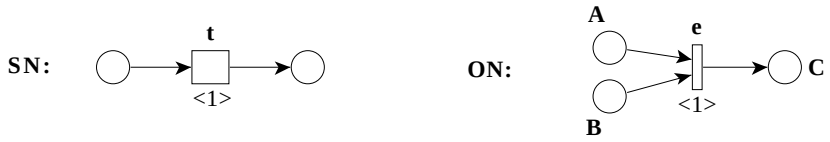

Figure 5.1: A SN transition interacting with an ON transition

The object system transitions from figure 5.1 are modelled by the LLPN transition in figure 5.2 with the guard formula $g(x, y) = \{\delta \otimes x \vdash y\}$ and the token formula $!(\delta \otimes A \otimes B \multimap C)$ for the object net transition.

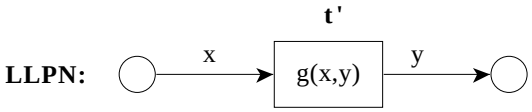

Figure 5.2: A LLPN equivalent to Fig. 5.1

In this example δ denotes a new propositional variable that is not otherwise used in the model. Unfortunately this is not sufficient to grant that the transition is fired synchronously to the required derivation in the token formula. Though the token formula can only transform A into B in the presence of δ which is only possible when executing the derivation forced by the guard of transition t'. On the other hand the guard formulae can be derived in Linear Logic calculus even if the token object residing in the input place and bound to x does not consume δ in the derivation of y, i.e. δ might still be present as a factor in the tensor product y if there is a derivation $x \vdash y'$ such that $y = \delta \otimes y'$.

To avoid this undesirable behaviour some extra measures have to be enforced. A possible solution is to modify the guard formula to

$$\delta \otimes \Gamma \otimes !(\delta \otimes \Sigma \multimap \Delta) \ \vdash \ (\Gamma - \Sigma + \Delta) \otimes !(\delta \otimes \Sigma \multimap \Delta)$$

where the capital greek letters denote multisets treated as formal sums. In addition it has to be checked that the arc variables x and y are bound to $\Gamma \otimes !(\delta \otimes \Sigma \multimap \Delta)$ and $(\Gamma - \Sigma + \Delta) \otimes !(\delta \otimes \Sigma \multimap \Delta)$, respectively[2], with Γ, Σ and Δ consistently bound to subformulae of x and y. This kind of guard formula prevents δ from being unaffected by the derivation and thus forces synchronization of the transition occurrence and derivation step. There may still be many possibilities to choose from just as a system net transition can have a variety of interaction partners in Valk's interaction relation. The actual interaction partner is chosen nondeterministically.

5.1.2 Autonomous Behaviour

For object systems two kinds of autonomous behaviour can be distinguished: object autonomous transitions occurrences and system autonomous transition occurrences. The

[2] $\Sigma \multimap \Delta$ denotes some transition in the canonical formula of the net and Γ represents the remainder of the net.

latter are called transport in [Val96b] and can easily be modelled in LLPNs by transitions whose guard formula is the constant true, i.e. the guard formula is $X \vdash X$ and all arcs connected to the transition bear the inscription X only.

Object autonomous transitions on the other hand are modelled by autonomous deduction steps within a LLPN.

5.1.3 Object System Marking and Occurrence Rule

There are of course many different ways to define markings of object systems according to the intuition underlying the specific model. One such possibility—process markings and extendibility of the processes in the input places of a system net transition—is discussed in [Val96b]. The solution given there is feasible in some cases of modelling but poses problems in others. It seems that a better solution would be to allow for different modes of enablement. This could be achieved by parametrization of the transitions leading to different transition types dealing with different token types. For example it my be feasible to copy an application form (simply use a xerox machine) but it is not feasible to copy a human being or an animal, though both might be represented by a token net within an object system.

Nevertheless it is possible to simulate the object system behaviour as defined in [Val96b] by an LLPN with a slightly modified calculus. We will give an informal discussion of the simulation in the following, giving first with an example stating that the unmodified Linear Logic calculus is not sufficient to represent the *least upper bound criterion* of object systems.

Example. Consider the object system depicted in figure 5.3. It is easily verified that if t_1 fires (transport) there is a possibility that the to copies of the object net autonomously execute different transitions leading to the situation in figure 5.4. This situation clearly is a deadlock as there does not exist any upper bound to the two object net processes. Thus t_2 is never enabled.

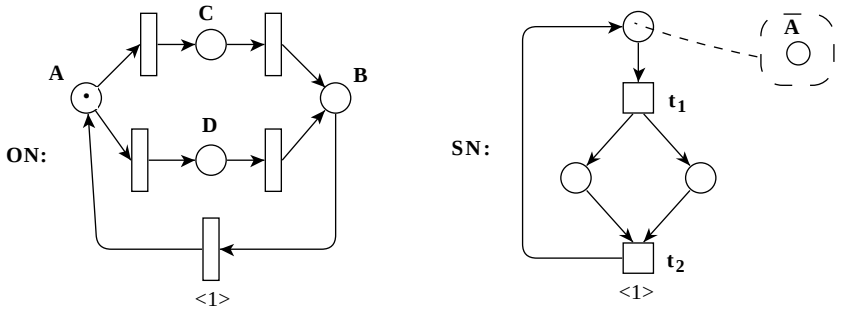

Figure 5.3: An object system with initial marking

This situation does not occur in the associated LLPN as there is no implicit information on the processes of the token net formulae. The object net marking in figure 5.4 would be described by the token formulae $C \otimes \varphi_{LL}(ON)$ and $D \otimes \varphi_{LL}(ON)$ respectively. But $B \otimes \varphi_{LL}(ON)$ is derivable from both such that t_2 would be enabled.

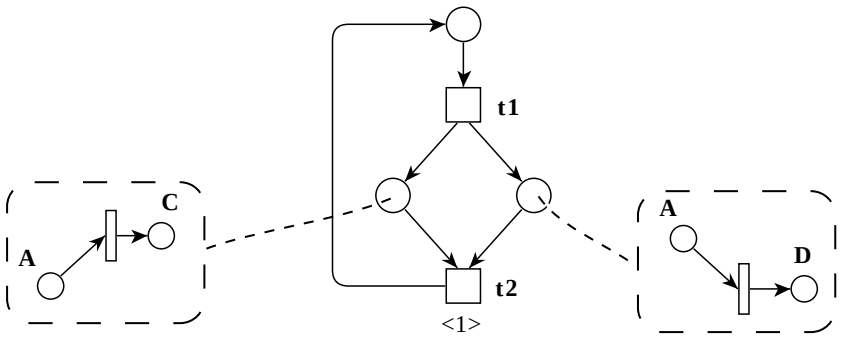

Figure 5.4: Deadlock situation for the net in Fig. 5.3

Of course it would be possible to incorporate more restrictions on the token formulae in the guard function of t_2. In the example above it would be sufficient to restrict the derivation to token formulae that both have resource C or both have resource D but this would lead to a kind of typed net[3], i.e. the system net assumes some information on the structure of the token.

Object Systems and Concurrency

In Valk's object systems there is a counter-intuitive restriction in the system behaviour, namely it is not possible for a transport (autonomous system net transition) and an autonomous object net transition to occur simultaneously. This problem is solved when using LLPN as a semantics for object systems: A derivation on the token formulae may very well be made while a transition fires, but is only required in the special case of synchronization. Thus LLPN allow for maximal concurrency.

5.2 Modifying Net Objects

Taking Petri nets as token objects displays some kind of dynamics within the tokens but there is a possibility of having even more dynamic tokens, namely tokens that change not only their state but also their structure. A structural change of an object net—which is not possible in object systems—could mean that some transition of the object net is removed, replaced, or added during the system net execution.

Example. Suppose it is possible to exchange some resource A for another resource B only once and that subsequent exchanges require to double the previously exchanged amount of As to receive still only one B. A possibility of modelling this situation is to modify the exchanging device/transition after each occurrence. The effect of such transition is shown in figure 5.5 where the grey object net is meant as the resulting marking after firing t.

This kind of behaviour can be expressed by the Linear Logic Petri net in figure 5.6 with the guard function $g(x_1, x_2, y_1, y_2) = [x_1, x_2, (\Gamma \multimap \Delta) \multimap (\Gamma^2 \multimap \Delta) \vdash y_1, y_2]$ if

[3] see section 5.3 for a discussion of typed LLPNs and actors

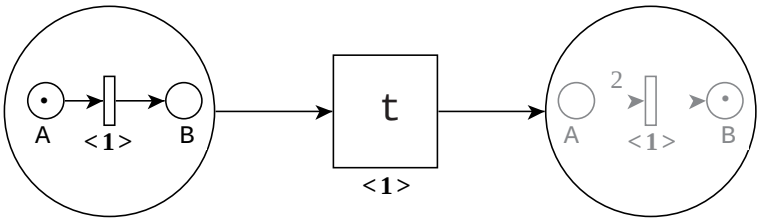

Figure 5.5: An system net modifying it's object net

$x_1 = \Gamma \multimap \Delta$ and $x_1 \neq y_1$. The object net systems are separated in this LLPN into a structural dynamic part (exchange rate) and a classic dynamic part (resources). Again the gray token formulae constitute the successor marking of the black formulae.

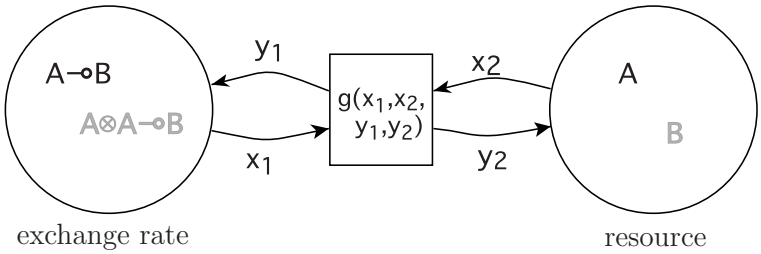

Figure 5.6: A LLPN for the net in Fig. 5.5

The transition takes a formula that represents a net and has a transition as substructure transforming it to another transition. In a similar fashion it is possible to check for the existence of any kind of subformula and simply consuming it by executing a transition. The conditional or unconditional generation/production of some subformula is also possible as discussed in section 5.3.1.

Remark. Note that the discussion above does not take into consideration what kind of structural changes may be desirable. This is left totally to the modeller. The only purpose for this section was to show that whatever restrictions might be imposed on the modifications, it is in principle possible to give a clear-cut semantics by means of Linear Logic Petri nets.

5.3 Agent Oriented Systems

Similar to the discussion in section 5.1 the token formulae that are consumed and produced by any transition can be required to have a specified appearance such as in-ports and out-ports for message passing. In this sense it is possible to have formulae denote actors and messages. This has been shown to be possible with Petri nets that have net systems (actors) with variable markings (messages) as arc inscriptions. These systems receive a precise semantics through LLPNs.

5.3.1 LLPN with occurrence check

In the previous section we have seen that pure logical derivations are not sufficient to model the exact behaviour of object systems. We have shown though that a simple equational theory suffices to give object systems a semantics in terms of Linear Logic Petri nets. For this purpose and for giving other formalisms a precise semantics we introduce a superclass of LLPN called OLLPN which adds some basic axioms for equivalence relations in order to make it possible to check for the occurrence of subformulae.

This is desirable, for instance, for agent oriented Petri nets or actor nets in which the token objects are actors/agents that pass messages in a predefined manner. We won't go into any details of features that agent oriented Petri net approaches have to include but we will show an example of such net. The example is only intended to motivate the necessity of the occurrence check feature in OLLPNs.

The main use of the occur check is in guaranteing that the actors that want to communicate have the general form that actors are supposed to have. This clearly includes the existence of input and output ports for the messages being communicated. The system net then has to implement the communication protocol. Thus in the following example the actors change their state according to the basic rules of communicating, e.g. a message has to reside in the output port of the sender before being sent and afterwards has to be in the in port of the receiver. The possibility of transmission errors is not taken into account in this simplistic example.

Example. Figure 5.7 shows on the left an agent net (an extension of coloured nets with nets as arc inscription) and on the right an OLLPN that gives the precise semantics if $g(X_1, X_2, Y_1, Y_2)$ has the following equations as condition

$$\begin{aligned} X_1 &= \langle N_1 \rangle \otimes IN_{\Gamma_1} \otimes OUT_{\Delta_1 \otimes m} \\ Y_1 &= \langle N_2 \rangle \otimes IN_{\Gamma_2} \otimes OUT_{\Delta_2} \\ X_2 &= \langle N_1{}' \rangle \otimes IN_{\Gamma_1} \otimes OUT_{\Delta_1} \\ Y_2 &= \langle N_2{}' \rangle \otimes IN_{\Gamma_2 \otimes m} \otimes OUT_{\Delta_2} \end{aligned}$$

and checks the derivability of $X_1, Y_1, IN_{\Sigma_1 \otimes m} \otimes OUT_{\Sigma_2} \multimap IN_{\Sigma_1} \otimes OUT_{\Sigma_2 \otimes m} \vdash X_2, Y_2$.

The capital greek letters are used for variables as before and $\langle N_i \rangle$ is used to denote the Linear Logic encoding of the remaining net system (including the current marking). The encoding used in this example makes use of the encoding for coloured Petri nets presented in section 3

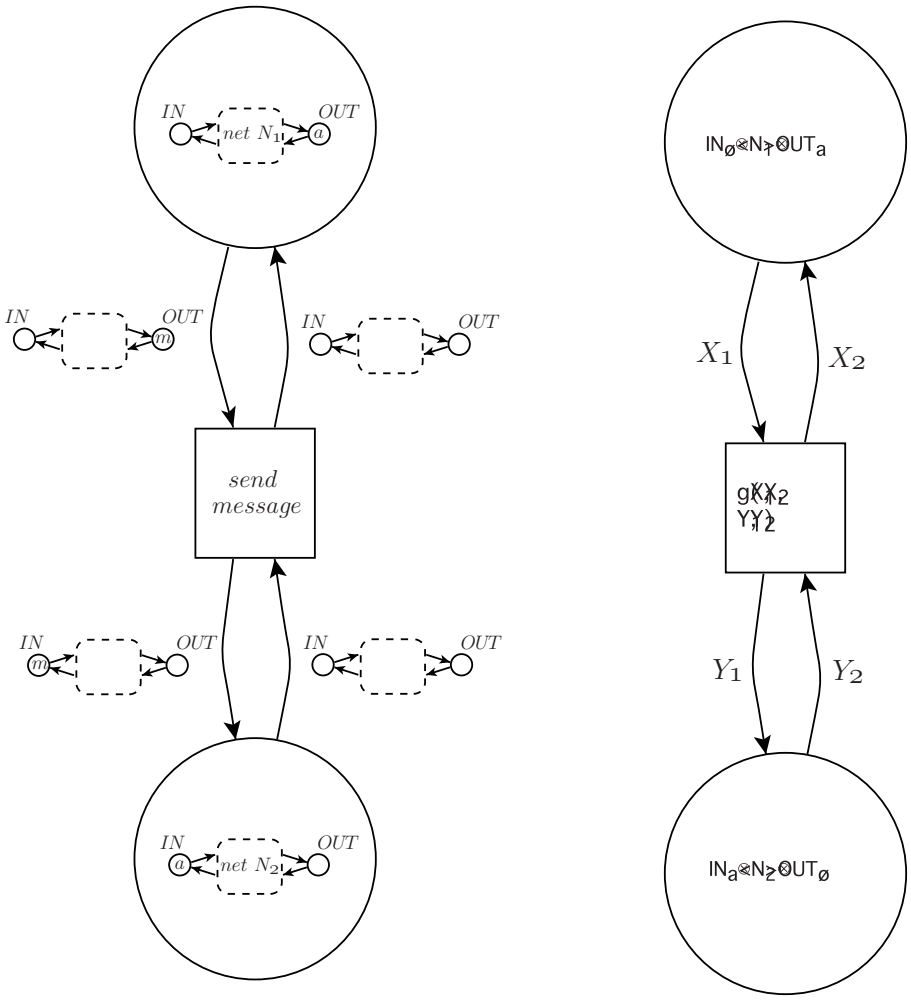

Figure 5.7: Agent oriented nets

Chapter 6

Further Results

In this section some further results are presented that are motivated by a Linear Logic perspective on Object Systems.

6.1 Nondeterministic Transitions

We will now consider an extended class of formulae compared with that studied in section 2.1. Allowing a very restricted use of the additive connective $\oplus$ we take into consideration derivations in the $(!, \oplus)$-Horn fragment of Linear Logic containing only $(!, \oplus)$-Horn sequents, i. e. only sequents of the following kind are allowed:

$$A_1 \otimes \cdots \otimes A_k, !\Gamma \vdash B_1 \otimes \cdots \otimes B_l$$

where the A_i and B_j are positive literals and Γ is a multiset of formulae of either of the two kinds (the C_i and D_j are also assumed to be positive literals):

- $C_1 \otimes \cdots \otimes C_n \multimap D_1 \otimes \cdots \otimes D_m$
- $C_1 \otimes \cdots \otimes C_p \multimap ((D_{1,1} \otimes \cdots \otimes D_{1,q_1}) \oplus \cdots \oplus (D_{r,1} \otimes \cdots \otimes D_{r,q_o}))$

The former kind of formula is exactly the one used to represent transitions in ordinary Petri nets. The latter can be used to represent nondeterministic transitions, i. e. transitions that have a set of sets of postconditions. In nets without capacity restrictions for places the enablement of a transition is defined for nondeterministic transitions in exactly the same way as in ordinary nets. The difference in firing such a transition is that there are many possibilities for the postset, one of which is chosen by the transition. This is viewed as internal nondeterminism of the net. An example of a nondeterministic transition is given in Figure 6.1 where the different postsets are marked by inscriptions $[i]$ and $[j]$ on the outgoing arcs.

In [Kan94] this kind of transition has been considered and the undecidability of the reachability problem for nondeterministic Petri nets is proved by reduction to vector games (a variation of vector addition systems).

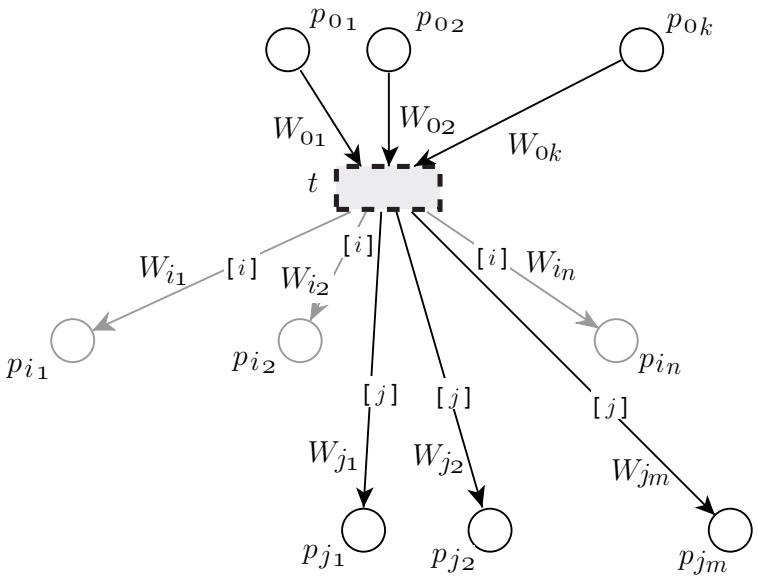

Figure 6.1: Example of a nondeterministic transition

Theorem 6.1.1 *The problem whether there exists a firing sequence in a nondeterministic Petri net that takes the initial marking* $\mathbf{m_0}$ *precisely to the marking* $\mathbf{m}$ *is undecidable.*

The previous result is used here to show that reachability in the object net is undecidable for generalised elementary object systems if a restriction on the system net process is considered. These are Petri net systems that consist of an elementary net system called system net, a P/T-net called the object net and an interaction relation between transitions of both nets. The tokens of the system net are defined to be the object net processes restricted by the interaction relation—demanding synchronization of certain pairs of transitions—and the firing rule (cf. [Val96b, Val96a, Val98b, Val95]). Object systems generalize and extend task/flow systems defined earlier in [Val91]. We give an outline of these constructions in the following paragraphs.

6.2 Generalised EOS

We generalise elementary object systems such that the object net may be an arbitrary P/T-net. The system net still has to be an elementary net system.

Definition 6.2.1 (Generalised EOS)
A generalised elementary object system is an object system that satisfies the following conditions:

1. *The system net* $\langle P_S, T_S, F_S, W_S, \mathbf{m}_S \rangle$ *is an elementary net system.*
2. *The object net* $\langle P_O, T_O, F_O, W_O, \mathbf{m}_O \rangle$ *is an ordinary P/T-net system.*

3. *The interaction relation $\rho \subseteq T \times E$ is an arbitrary relation between system and object transitions.*

The enablement of transitions is defined as in elementary object systems.

6.3 Decidability Issues

Whereas the reachability in P/T-nets is known to be decidable we show that reachability for some partial specifications of object systems, namely system processes, is undecidable.

6.3.1 Reachability in Object Systems

Definition 6.3.1 (Reachable Marking of a P/T-Net) *A marking m of a P/T-net $N = (P, T, W, M)$ is reachable from the initial marking M iff there exists $w \in T^*$, i.e. a finite occurrence sequence $w = t_1 t_2 \cdots t_n$ were $t_i \in T$ for all $i \in \{1, \ldots, n\}$, such that*

$$m_0 \rightarrow_{t_1} m_1 \rightarrow_{t_2} \cdots \rightarrow_{t_n} m_n,$$

with $m_0 = M$ and $m_n = m$.

Definition 6.3.2 (Reachability Problem for P/T-Nets) *Given an arbitrary marking of a P/T-net N, decide whether that marking is reachable from the initial marking of N. This is called the reachability problem for P/T-nets.*

Theorem 6.3.3 *The reachability problem for EOS is decidable.*

Proof Sketch. For every elementary object system EOS there exists a folding into a P/T-net N, and a bijective mapping between markings m_N of N and markings m_{EOS} of EOS, such that m_N is reachable in N iff m_{EOS} is reachable in EOS. As it is a well-known fact that reachability for P/T-nets is decidable we can use this result to show the decidability of the reachability problem for elementary object nets. The best upper bound that has been found for the complexity of the reachability problem in P/T-nets is not even a primitive recursive function (actually a double-exponential function in the net size). □

Theorem 6.3.4 *The reachability problem for generalised elementary object systems relative to the system net occurrence sequences is undecidable.*

Proof. We show that nondeterministic (sometimes called generalised) Petri nets are reducible to generalised elementary object systems such that the admissible processes of the nondeterministic net are exactly the admissible systems of the object system.

Note that for any transition t of a nondeterministic Petri net its postset $t^\bullet$ is a set, i.e. $t^\bullet \subseteq \mathrm{pow}(P)$, where $\mathrm{pow}(P)$ denotes the powerset of P. If $\mathrm{card}(t^\bullet) > 1$ then t is a nondeterministic transition, otherwise t is an ordinary transition.

Nondeterministic Petri nets have an undecidable reachability problem as proved by Kanovich in [Kan94]. Thus reachability in GEOS is also undecidable for any given system net process.

The simulating GEOS will be built of a system net which consists of only one place connected to transitions named exactly like all those of the nondeterministic Petri net. The firing of these system net transitions will be restricted by the interaction relation that allows for a choice of transitions in case the original transition was nondeterministic. The nondeterministic transition from Figure 6.1 would be simulated by the GEOS from Figure 6.2. Nondeterministic transitions are shown as solid lines whereas normal transitions are Boxes. The interaction relation is indicated by $\langle 1 \rangle$, meaning that the system net transition must interact with one of the object net transitions marked by the same number. Both partial constructions must then be applied to all remaining transitions of the nondeterministic net.

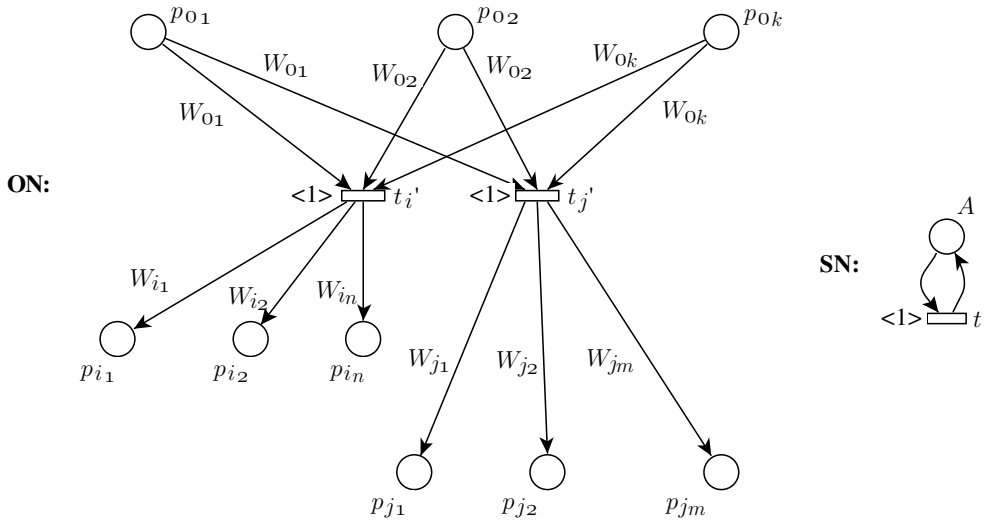

Figure 6.2: Example of a nondeterministic transition simulated by a GEOS

We will now formalise the aforementioned reduction:

To construct an equivalent generalised elementary object system $OS = (SN, ON, \rho)$ for a given nondeterministic Petri net system $\mathcal{S}(\mathbf{m}) = \langle \mathcal{N}, \mathbf{m} \rangle$ with sets of places P and transitions T, we have to construct a system net SN, an object net ON, and an appropriate interaction relation ρ. We assume without loss of generality that for all $t \in T$ the set of postconditions is $t^\bullet = \{X_0, \ldots, X_{n_t}\}$, where $X_i \in \mathrm{pow}(P)$ with associated weight functions $W_i(t,p)$ for all $p \in X_i$, $i \in \{0, .., n_t\}$ and $n_t := \mathrm{card}(t^\bullet)$.

1. $SN = \langle \{A\}, T, F_S, W_S, \mathbf{m}_S \rangle$, with $\mathbf{m}_S[A] = 1$, and $\forall t \in T. W_S(A,t) = 1 \wedge W_S(t,A) = 1 \wedge (A,t) \in F_S \wedge (t,A) \in F_S$

2. $ON = \langle P, T_O, F_O, W_O, \mathbf{m} \rangle$, with $T_O \{t_i' \mid t \in T \wedge i \in \{1, \ldots, \mathrm{card}(t^\bullet)\}\}$, $\forall t \in T\,.\,\forall i \in \{0, .., n_t\}\,.\,p \in {}^\bullet t \rightarrow (p, t_i) \in F_O \wedge W_O(p,t_i) = W\,(p,t)$ and $\forall t \in T. \forall i \in \{0, .., n_t\}. \forall p \in X_i. (t_i, p) \in F_O \wedge W_O(t_i, p) = W_i(t,p)$.

3. $\rho = \{(t, t_i') \mid t \in T \wedge i \in \{1, \ldots, \mathrm{card}(t^\bullet)\}\}$

It is left to the reader to show formally that the set of system net occurrence sequences of OS determined by the set of SN-processes for OS, is equal to the set of occurrence sequences of $\mathcal{S}$, and that the resulting markings of N and ON correspond to each other w.r.t. the simulation relation defined above.

Clearly for each possible postset of any nondeterministic transition in $\mathcal{S}$ there exists exactly one transition in the object net of the corresponding object system OS. On the other hand there are no other alternative transitions to choose from unless there is also a choice of which nondeterministic transition to fire in $\mathcal{S}$. The reverse direction is also true, i.e. whenever some system net transition is enabled according to the firing rule of object nets the corresponding nondeterministic transition is also enabled if we transfer the object net marking to $\mathcal{S}$.

□

6.3.2 Reachability in LLPNs

The argument of undecidability of the reachability problem discussed in the preceding section applies also to the general class of Linear Logic Petri nets as follows.

Theorem 6.3.5 *The reachability problem for general Linear Logic Petri nets is undecidable.*

Proof. Taking as token formulae exactly that fragment of Linear Logic studied in [Kan94] we see that the reachability problem for LLPNs is undecidable by a trivial construction of an LLPN. Details are left to the reader.

Also most fragments of Linear Logic are undecidable. This makes the reachability problem for LLPNs based on such fragment undecidable. □

Chapter 7

Bibliographic Remarks

Introductions to Linear logic apart from Girard's original paper [Gir87] are available in [Sce93],[Tro93],[Tro92], and [Wad93]. For a thorough treatment of decidability and complexity issues of the various fragments of Linear Logic we refer the reader to [LSS92]. An abridged version of this paper has been published as [Far98]

Bibliography

[Bro89] C. Brown. Relating Petri Nets to Formulae of Linear Logic. Technical report, Department of Computer Science, University of Edinburgh, 1989.

[EW90] U. Engberg and G. Winskel. Petri Nets as Models for Linear Logic. Technical report, Computer Science Department, Aarhus University, 1990.

[Far96] B. Farwer. Relating object systems to formulae of infinitary linear logic. Talk given at the Third Seminar on Algebra, Logic, and Geometry in Informatics (ALGI 3), Tokyo, 1996.

[Far98] B. Farwer. A Linear Logic View of Object Systems. In *Concurrency Specification and Programming. Proceedings.* Humboldt-Universität, Berlin, 1998.

[Gen35] G. Gentzen. Untersuchungen über das logische Schliessen. *Mathematische Zeitschrift*, 39:176–210, 405–431, 1935.

[Gir87] J.-Y. Girard. Linear Logic. *Theoretical Computer Science*, 50:1–102, 1987.

[Jen92] K. Jensen. *Coloured Petri Nets - Basic Concepts, Analysis Methods and Practical Use, Vol. 1: Basic Concepts.* EATCS Monographs in Theoretical Computer Science. Springer-Verlag, 1992.

[Kan94] M. I. Kanovich. Petri Nets, Horn Programs, Linear Logic, and Vector Games. In M. Hagiya and J. C. Mitchell, editors, *Theoretical Aspects of Computer Software, International Symposium TACS'94, Sendai, Japan*, LNCS 789, pages 642–666. Springer-Verlag, 1994.

[KI97] M. I. Kanovich and T. Ito. Temporal Linear Logic Specifications for Concurrent Processes. In *Logic in Computer Science (LICS). Proceedings.* IEEE Comput. Soc. Press, 1997.

[LSS92] P. Lincoln, A. Scedrov, and N. Shankar. Decision Problems for Propositional Linear Logic. *Annals of Pure and Applied Logic*, 56:239–311, 1992.

[MOM89] N. Martí-Oliet and J. Meseguer. From Petri Nets to Linear Logic. Technical report, SRI International, Computer Science Laboratory, Stanford, 1989.

[Sce90] A. Scedrov. A Brief Guide to Linear Logic. *Bulletin of the EATCS*, 41:154–165, 1990.

[Sce93] A. Scedrov. A Brief Guide to Linear Logic. In G. Rozenberg and Arto Salomaa, editors, *Current Trends in Theoretical Computer Science*, pages 377–394. World Scientific, 1993. Revised version of [Sce90].

[Tan97] M. Tanabe. Timed Petri Nets and Temporal Linear Logic. In *Applications and Theory of Petri Nets 1997. Proceedings*, LNCS 1248. Springer-Verlag, 1997.

[Tro92] A. S. Troelstra. *Lectures on Linear Logic*. CSLI Lecture Notes No. 29, Stanford, 1992.

[Tro93] A.S. Troelstra. Tutorial on Linear Logic. In Peter Schröder-Heister and Kosta Došen, editors, *Substructural Logics. Workshop Tübingen 1990*. Oxford Science Publications, 1993.

[Val91] R. Valk. Modelling Concurrency by Task/Flow EN Systems. In *3rd Workshop on Concurrency and Compositionality*, number 191 in GMD-Studien, St. Augustin, Bonn, 1991. Gesellschaft für Mathematik und Datenverarbeitung.

[Val95] R. Valk. Petri nets as dynamical objects. In *Workshop Proc. 16th International Conf. on Application andTheory of Petri Nets, Torino, Italy*, June 1995.

[Val96a] R. Valk. How to Define Markings in Object Systems. *Petri Net Newsletter 50*, 1996.

[Val96b] R. Valk. On Processes of Object Petri Nets. Technical report, FBI-HH-B-185/96, Fachbereich Informatik, Universität Hamburg, 1996.

[Val98a] R. Valk. Introduction to Petri Nets. Lecture Notes of the MATCH summer school 1998. System Engineering: A Petri Net Based Approach to Modelling, Verification and Implementation, 1998.

[Val98b] R. Valk. Petri Nets as Token Objects. An Introduction to Elementary Object Nets. In J. Desel and M. Silva, editors, *Applications and Theory of Petri Nets 1998. Proceedings*, LNCS 1420. Springer-Verlag, 1998.

[Wad93] P. Wadler. A Taste of Linear Logic. In *Mathematical Foundations of Computer Science*, pages 185–210. Springer-Verlag, 1993.