Chapter 0

Introduction

In 1962 C.A. Petri presented the foundations for a new theory for describing information flows [74, 75]. These foundations were based on both theoretical and practical considerations. In contrast to the classical automata theory, where the states are *global* and a transition leads from one global state to another, in this theory states are distributed over *local* states and a transition alters several of these local states. A central idea of this theory is, that the effect of transition occurrences is purely local and that no assumptions are made with respect to the existence of a global control. Consequently, the theory is truly non-sequential.

This theory resulted in a model based on *nets*. These nets are bipartite graphs, obtained by viewing states and transitions as dual concepts. In this model of *Condition/Event systems* (*C/E systems*), the local states, called *conditions*, can have two values. C/E systems, or actually several variants of them, have a rich semantic theory for describing their behaviour at several levels of abstraction. For instance, transition systems, trace languages, and event structures have been used to represent their behaviour from different points of view.

Already at an early stage in the investigation of C/E systems an important generalization of this model was proposed in which the local states can have a finite but arbitrary number of values rather than only two. This generalization was called *Place/Transition systems* (*P/T systems*). Initially, the main interest in these objects was due to the fact that they were recognized to be mathematically equivalent to the model of *Vector Addition Systems*, discovered independently by Karp and Miller in connection with their work on parallel program schemata [49]. Since then, the model of P/T systems has turned out to be a model for concurrent systems which is in several respects a more attractive model than C/E systems. The model of P/T systems is the main subject of study in this thesis. From now we refer to this model simply as *Petri nets*.

The aim of this thesis is to investigate ways to describe the behaviour of Petri nets, which is not as well-understood as that of C/E system. To achieve this we investigate conservative extensions of the transition system semantics, the trace semantics, and the event structure semantics of (variants of) C/E systems. We hope that in this way we contribute to a better insight in the aspects of concurrency which play a significant role at the level of Petri nets.

In this introduction we first discuss in Section 0.1 the model of Petri nets. One can take several points of view in analyzing the behaviour of these Petri nets. They are presented in Section 0.2. In Section 0.3 we then discuss how the language of category theory can be used to study the relationship between Petri nets and various models used for describing the behaviour of Petri nets. Finally, in Section 0.4 we give some historical background of the theory of Petri nets, and in Section 0.5 we outline the contents of this thesis.

0.1 Petri Nets

The Petri net model is operational in the sense that it is based on the notions of *states* and *transitions* between states. This is similar to the approach in the classical sequential automata theory. In Petri nets however, (global) states are distributed over "local" states. These local states are represented as tokens in the *places* of the Petri net. Each transition is connected to certain input and output places according to a fixed directed and weighted neighbourhood relation. Then a transition occurrence is purely local: only those places to which a transition is connected are involved.

In Petri nets, places may contain multiple, indistinguishable tokens (which may be viewed as available resources), and transitions are not labelled. The dynamic behaviour of Petri nets is given by the firing rule. This firing rule specifies that transitions can occur if the input places of these transitions contain "enough" tokens, and that the effect of transition occurrences consists of removing some tokens from the input places and putting some tokens into the output places.

The Petri net in the following standard example illustrates this firing rule.

Example 0.1.1

In Figure 0.1 a Petri net is given which models a system with a producer and a consumer. The producer repeatedly produces two items and then puts them in a buffer, which can contain an arbitrary, finite number of items (represented by tokens). The consumer repeatedly removes one item from the buffer and then consumes this item. In the initial state of the system, called the *initial marking*, places p_1 and c_1 both contain one token and the other places are empty. At the initial marking, the transition *produce* can occur. If it occurs, it removes a token from place p_1, and puts two tokens into place p_2. At the initial marking also the transition *stop* can occur. As p_1 contains only one token, the transitions *produce* and *stop* cannot both occur, although each of them can occur individually. After the transition *produce* has occurred, the transition *put in buffer* can occur, which has the effect that two tokens are removed from place p_2, two tokens are put in place *buffer*, and one token is put in place p_1. In the resulting marking the transitions *produce* and *remove from buffer* can occur. In fact, they can occur together, because each place contains enough tokens for both transitions.

Finally observe that there is no upper bound on the number of tokens that can be put in place *buffer*, because the producer can repeatedly put items into the buffer. □

Several of the notions considered in this thesis have been developed first in the context of variants of the original model of C/E systems such as *elementary net systems* [81, 88, 80, 89] and the net systems from [57, 98] which we call here *safe net systems*.

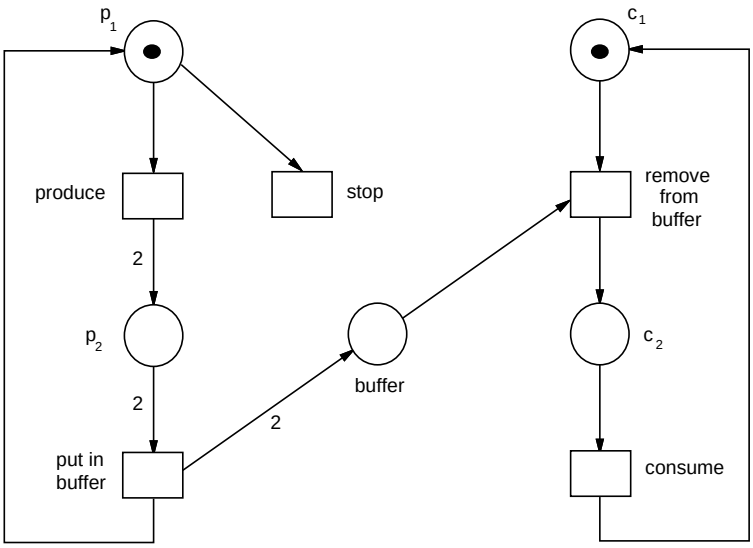


Figure 0.1: A producer and a consumer

In the terminology used for these net systems, places are called *conditions*, transitions are called *events*, and markings are called *cases*. Actually, markings which assign more than one token to a place are not considered in the approach followed for these net systems, and so cases are sets of conditions (which contain a token). Also, a different firing rule is used. Rather than inspecting only the input conditions to see if an event can occur, now also the output conditions must be inspected. In an elementary net system an event can occur iff each of its input conditions contains a token *and* all of its output conditions are empty. The effect of an occurrence of an event is the same as for Petri nets: the tokens from the input conditions are removed and tokens are put into the output conditions. For safe net systems this firing rule is relaxed by allowing an event to occur if all its input conditions hold and the only output conditions that are not empty are those that are also input conditions.

Elementary net systems and safe net systems are closely related to *1-safe* Petri nets. A Petri net is 1-safe if it has the property that no execution of it according to the firing rule leads to a place with two or more tokens. As far as the dynamic behaviour is concerned, elementary net systems and safe net systems which are *contact-free* can be viewed as 1-safe Petri nets [81, 98]. Elementary net systems and safe net systems are called contact-free if they have the property that the holding of the input conditions of an event guarantees that the event can occur. Thus contact-free elementary net systems can be considered as a subclass of 1-safe Petri nets, while the class of contact-free safe net systems can be identified with the class of 1-safe Petri nets.

For the behavioural notions considered in this thesis, for every elementary net system or safe net system there exists a contact-free elementary net system or contact-free safe net system, respectively, which yields the same behaviour [81, 98]. For the sake of uniformity in the presentation we will mainly consider 1-safe Petri nets rather than (contact-free) elementary net systems or (contact-free) safe net systems in this

thesis.

Observe that elementary net systems and safe net systems are defined through their firing rule, whereas the definition of 1-safe Petri nets is behavioural, i.e. given in terms of *reachable* markings.

The models of 1-safe Petri nets, elementary net systems, and safe net systems have a very rich and elegant theory for analyzing their behaviour (see, e.g., [78, 67, 98]). It seems however that in order to lift this theory to the level of arbitrary Petri nets, much more problematic aspects of concurrency must be confronted. On the other hand, there are several reasons why Petri nets form an attractive model to work with.

First of all, the model of Petri nets is mathematically equivalent to the model of *Vector Addition Systems* [49] which has been discovered independently of Petri nets. These Vector Addition Systems, and hence also Petri nets, give rise to some interesting decision problems [49]. Using a similar idea, Petri nets can also be viewed as simple examples of *multiset rewrite systems* which have gained a lot of attention recently. From this viewpoint a Petri net is a set of places together with a set of rewrite rules (i.e. transitions). Each rewrite rule specifies how a multiset of places (the input places of the transition) can be rewritten to another multiset of places (the output places of the transition).

In the programming language Γ of Banâtre and Le Métayer [4] multisets are proposed as a suitable datastructure for parallel programming. A Γ program consists of rewrite rules where each rewrite rule has an associated condition. The existence of those conditions is the main difference with Petri nets. If a multiset of data satisfies the condition associated with a certain rewrite rule for this multiset, then a "chemical reaction" can take place which replaces the multiset according to this rewrite rule. The Γ language treats parallellism at the *logical* level, because no global control is specified over the actions which are performed. This is quite different from several conventional parallel programming languages where arrays are used as the main datastructure and a global control is specified for the actions, so that parallellism is treated at the *implementation* level.

The *Chemical Abstract Machine* (*CHAM*) of Berry and Boudol [7] is also based on the idea of multiset rewriting. One of the extra features of the CHAM is that it has a "membrane" construct allowing to "encapsulate" parts of the data. The CHAM can be used for describing the operational semantics of process calculi such as (a fragment of) Milner's π-calculus [62]. Also in [25] multiset rewrite systems (in fact Petri nets) are used for describing the operational semantics of a fragment of the π-calculus.

A second reason why Petri nets form an attractive model to work with is that they have a very smooth algebraic structure. A common way, see, e.g., [78], to describe the effect of transition occurrences in a Petri net N on the number of tokens in each place is to use a matrix representation $\underline{N}$ of N. Consider, e.g., the Petri net depicted in Figure 0.1. Then this Petri net has the following matrix representation.

	stop	*produce*	*put in buffer*	*remove from buffer*	*consume*
p_1	-1	-1	1	0	0
p_2	0	2	-2	0	0
c_1	0	0	0	-1	1
c_2	0	0	0	1	-1
buffer	0	0	2	-1	0

Such a matrix representation of a Petri net is useful for analyzing certain properties of the Petri net. In particular, this matrix representation is used for determining the *S-invariants* and *T-invariants* of a Petri net N [51, 78]. S-invariants are solutions x of the equation $\underline{N}^T \cdot x = 0$ (where $\underline{N}^T$ is the transpose of $\underline{N}$) and T-invariants are solutions y of the equation $\underline{N} \cdot y = 0$. S-invariants represent those linear combinations of places for which the number of tokens remains constant. They are useful for, e.g., analyzing liveness and safeness properties of the Petri net. T-invariants on the other hand represent linear combinations of transitions the occurrence of which does not change the original marking. For instance, the S-invariants of the Petri net depicted in Figure 0.1 are all vectors $\lambda \cdot \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}$ with $\lambda \in \mathbf{N}$. This implies that the number of tokens in c_1 and c_2 (which is 1 initially) remains constant, and no other linear combination of places has a constant token count. The T-invariants of this Petri nets are all vectors $\lambda \cdot \begin{pmatrix} 0 \\ 1 \\ 1 \\ 2 \\ 2 \end{pmatrix}$ with $\lambda \in \mathbf{N}$. Thus, if in an arbitrary marking the transitions *produce* and *put in buffer* both occur an equal number of times and the transitions *remove from buffer* and *consume* both also occur an equal number of times which is twice the number of occurrences of *produce*, then the resulting marking is the same as the original one.

The algebraic nature of Petri nets is used explicitly by Winskel in [95]. There it was observed that a Petri net is essentially a 2-sorted algebra, where one sort is the set of multisets over places and the other sort is the set of multisets over transitions. This algebra has three operations: one constant which gives the initial marking, a matrix which gives the number of tokens removed by each transition from each place, and a matrix which gives the number of tokens put by each transition in each place. Thus these last two operations map multisets of transitions to multisets of places and the subtraction of the two matrices is the matrix $\underline{N}$ described above. One of the motivations for this view of Petri nets is that it leads to a natural notion of morphisms between Petri nets which is discussed later in this introduction.

In a similar fashion, also Montanari and Meseguer have exploited the algebraic nature of Petri nets [59]. They made use of the fact that a Petri net can be viewed as a graph, the nodes of which are the multisets over the places, and the arrows of

which are the transitions of the Petri net, where the source and target of an arrow are multisets over the input- and output places of the transition, respectively. Actually, this is not an ordinary graph, because the set of nodes of the graph has in addition a (free) monoid structure (with multiset addition as its operation and the empty multiset as its unit). Also this view of Petri nets gives rise to a natural notion of morphisms between Petri nets. By defining an operation of parallel composition on the arrows, the monoid structure on the nodes can be extended to the arrows. In this way a Petri net itself may be viewed as a monoid. An interesting consequence of the algebraic view proposed by [59] is that it allows to treat Petri net computations at several levels of abstraction in an algebraic way.

A third reason why Petri nets form an attractive model to work with, is that they are a more powerful tool for modelling systems than 1-safe Petri nets. As such they also form the basis for the model of *Coloured Petri nets* from Jensen which is used in several industrial applications [47].

Finally, Petri nets seem to have a natural relationship to linear logic [31, 23]. This relationship is as yet not completely worked out in a satisfactory manner. It is our hope however that a deeper understanding of the behaviour of Petri nets will contribute to the formulation of a model theory for linear logic in terms of Petri nets.

0.2 The Behaviour of Petri Nets

Petri nets are very concrete objects. In order to analyze their behaviour, it is useful to get rid of details by relating them to more abstract models of concurrency. In this way it becomes possible to identify Petri nets which have essentially the same behaviour. This is useful for, e.g., defining a notion such as place refinement [69]. More abstract models may also serve as an intermediate step for obtaining a Petri net semantics for process algebras such as CCS [65].

For 1-safe Petri nets, or actually variants of C/E systems such as (contact-free) elementary net systems and (contact-free) safe net systems, several models have proved to be very successful for describing their behaviour. Roughly, a distinction can be made between linear time models, in which non-conflicting (initial parts of) "runs" of a concurrent system are represented, and branching time models where also conflicts are taken into account. Thus for the linear time models there are several objects associated with one 1-safe Petri net, each corresponding to a different way in which conflicts are resolved in (an initial part of) a run.

A linear time semantics for 1-safe Petri nets or elementary net systems has been proposed via its non-sequential *processes* [76, 11, 67]. These processes are based on (labelled) *causal nets* and represent the runs of 1-safe Petri nets (or elementary net systems). Causal nets are acyclic nets in which the places are non-branching. As a consequence, no *forward* or *backward conflicts* occur. Thus causal nets induce a partial ordering relation over its places and transitions which, when restricted to the transitions, leads to a partial order description of the transition occurrences in the original Petri net. The corresponding reachable markings of the original Petri net are represented as slices, i.e. maximal unordered sets of places, in the causal net.

The branching aspects of a 1-safe Petri net, while still maintaining the distribution of a state over local states, can be represented by relaxing the conditions on causal nets to allow forward conflicts, leading to the notion of *occurrence nets* [66, 96]. In this way also initial parts of "branching runs" can be represented, the resulting objects of which are called branching processes in [24]. With this approach it is now however also possible to capture the behavioural aspects of a 1-safe Petri net in a *single* object, its unfolding [66]. A transition in the unfolding can be viewed as corresponding to the "event" of the occurrence of a transition in the Petri net.

In the two approaches mentioned above, the model describing the behaviour of a 1-safe Petri net still has a notion of a distributed state. Abstracting in a Petri net from the distribution of a global state over local states leads to a *transition system* model. Transition systems are obtained from nets by the classical case graph/marking diagram construction, see, e.g., [78]. For elementary net systems it is sufficient to consider sequential transition systems because the concurrency in the net can be recovered from the sequential case graph by certain diamond properties [39]. In [68] a characterization is given of the transition systems, called elementary transition systems, obtained in this way. For 1-safe Petri nets and safe net systems however, these diamond properties do not apply, so that more structure in the transition system is necessary to represent concurrency. In this case it is sufficient to work with *asynchronous transition systems* [6, 86]. In this model concurrency is represented explicitly by a (global) binary independence relation over actions. A characterization of the asynchronous transition systems corresponding to 1-safe Petri nets is given in [98].

A linear time semantics for 1-safe Petri nets while abstracting from the distribution of a global state over local states can be given by using Mazurkiewicz' *trace theory* to represent the non-conflicting runs of a 1-safe Petri net [55, 56]. Here the concurrency present in the net is also formalized in a binary independence relation over its actions, i.e. transitions. This independence relation induces an equivalence relation over the sequential runs (firing sequences) of the net. Each of the resulting equivalence classes, which are called traces, represents a single non-sequential run.

Mazurkiewicz' trace theory has turned out to be a model for concurrency which is also of independent interest, see, e.g., [20, 56, 1]. On the one hand it is a language-based model. On the other hand trace theory can also be studied algebraically in the context of free partially commutative monoids [8, 28]. Furthermore, traces also admit a graphical representation in terms of independence graphs, thus supporting the claim that it is a partial order model for concurrency (see, e.g., [56]).

As a branching time semantics model for the behaviour of 1-safe Petri nets which abstracts from the distribution of a global state over local states, *prime event structures* have been proposed [66]. Here the relationship between the transition occurrences in a 1-safe Petri net, called *events*, is described by both an ordering relation and a binary conflict relation over the events. Prime event structures also occur as concrete representations of certain Scott Domains [85]. In this way [66] provides a connection between 1-safe Petri nets and domain theory.

The models mentioned above are summarized in the following diagram.

		linear time	*branching time*
distributed state	1-safe Petri nets	causal nets	occurrence nets
global state	asynchronous transition systems	Mazurkiewicz' trace languages	prime event structures

The models in the first row have a notion of a distributed state, whereas the models in the second row have only a notion of a global state. From left to right the models are increasingly more abstract. It turns out that the different types of semantics are closely related. For the relationships between the various models mentioned above, see, e.g., [66, 67, 84, 70, 98].

An interesting question that now arises is, to what extent the above approaches can be lifted to arbitrary Petri nets. This would then lead to a description of the behaviour of Petri nets from different points of view. It turns out however, that for Petri nets much more problematic aspects of concurrency must be dealt with.

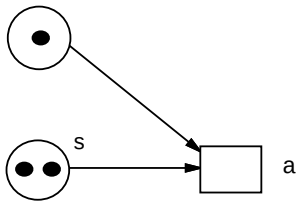


Figure 0.2: A Petri net in which a can occur only once

Also for arbitrary Petri nets, non-sequential processes based on causal nets have been proposed as a description of their behaviour [34, 9, 11]. The notion of non-sequential processes is based on the intuition concerning the behaviour of 1-safe Petri nets by viewing the tokens in a place more or less as "coloured", i.e. distinguishable, entities. Consider, e.g., the Petri net depicted in Figure 0.2. In this Petri net the transition a can occur only once. However, by viewing the tokens as distinguishable entities a distinction is made between the occurrence of a with the left token in s and the occurrence of a with the right token in s. Such a colouring of tokens is not very satisfactory because it destroys the possibility of viewing Petri nets as simple *multiset* rewrite systems. It also leads to the counter-intuitive result that 1-safe Petri nets and arbitrary Petri nets give rise to the same set of behaviours. To deal with these problems, an equivalence relation over processes is defined in [9] in order to identify processes which differ only in the colouring of the tokens.

An alternative description of the runs of a Petri net is given in [59, 18]. Based on the graph representation of a Petri net, net computations are defined as the arrows obtained by closing the arrows of the graph with respect to operations of parallel and sequential composition. Some natural axioms are then defined to identify "equivalent" computations. It turns out that the equivalence classes obtained in this way correspond exactly to the equivalence classes of processes from [9].

With respect to the branching time models, occurrence nets have been used [33, 35, 24, 60] for defining also the unfolding of not necessarily 1-safe Petri nets. In

[60, 61] the more sophisticated notion of decorated occurrence nets is introduced. These approaches again lead to an event structure semantics for Petri nets in terms of prime event structures [24, 60]. In all these approaches for giving a branching time semantics for Petri nets however, the tokens are treated again as coloured entities.

The aim of this thesis is to give a proper generalization of the second row in the diagram when going from 1-safe Petri nets to arbitrary Petri nets. Thus we look for a transition system semantics, a trace semantics, and an event structure semantics for Petri nets. By thus abstracting from the distribution of a global state over local states, it seems easier to avoid assumptions involving some kind of colouring of the tokens.

The behaviour of Petri nets in terms of transition systems is given by its marking diagram, see, e.g., [78], which is a transition system in which the transitions are labelled with multisets of concurrently occurring transitions of the Petri net. A characterization of the transition systems associated with Petri nets has been given in [63].

The semantics for 1-safe Petri nets in terms of Mazurkiewicz' traces cannot be lifted directly to the model of arbitrary Petri nets. The problem is that because a place may now contain many tokens, concurrency (and conflict) are no longer global structural relations, but depend on the current marking. Moreover, concurrency between transitions at a marking can no longer be characterized through a binary relation.

To deal with these problems, we introduce *local trace languages*. These local trace languages generalize Mazurkiewicz' trace languages along three dimensions. Firstly, we consider *multiset sequences* instead of ordinary sequences. Secondly, we consider independence relations that are context-dependent, where the context is defined by a multiset sequence. Thirdly, we specify in the independence relation a finite multiset of actions that can occur concurrently at a context rather than just a pair of symbols that may commute as in the classical case. It is then straightforward to lift the standard notions from trace theory to the, much richer, new setting. It turns out that the trace semantics for Petri nets obtained in this way agrees with the semantics in terms of equivalence classes of processes and of equivalence classes of occurrence sequences from [9].

One of the advantages of our trace semantics for Petri nets is, that it also serves as a basis for a branching time semantics for Petri nets in terms of event structures. In order to give such a branching time semantics for Petri nets, we propose a generalization of the prime event structure semantics for 1-safe Petri nets with the help of a new class of event structures, called *local event structures*. These event structures are easy to define and require just a purely local concurrency axiom; no global order theoretic properties are demanded. It turns out that a subclass of local event structures can be advocated as a partial solution to the problem of identifying the event structures that correspond to the behaviour of Petri nets. The solution is partial in that in the event structure semantics for Petri nets that is being proposed here, auto-concurrency is filtered out from the behaviour of Petri nets. Auto-concurrency is the phenomenon by which multiple instances of a transition become enabled at a marking. This is impossible in a 1-safe Petri net. Even though our event structure semantics is restricted in this sense, the event structure semantics is a non-trivial proper extension of the prime event structure semantics for 1-safe Petri nets. Moreover, our event structure semantics does not assume any colouring of tokens, in contrast to the branching time semantics for

Petri nets via (decorated) occurrence nets and prime event structures mentioned above.

0.3 The Categorical Approach

Given a semantic model to be used for representing the behaviour of Petri nets, a semantic map associates a behavioural object from the chosen model with each Petri net. An obvious question that arises is, how to show that such a map is the "right" one. As we argue in this section, the language of *category theory* (see, e.g., [53, 5, 77]), and in particular the notion of an *adjunction*, is ideally suited for this purpose.

The models for concurrent systems that we consider are meant to model dynamic systems. Hence it is reasonable to equip the various models with (structure-preserving) morphisms to capture the fact that within a model one system is capable of simulating another system. Defining a composition operation for morphisms and an identity morphism for each object leads then to a representation of a model as a category.

The models we consider are at different levels of abstraction. With each object in a concrete model a semantic map associates an object in an abstract model by "forgetting" some of the structure. Now in order to compare the two models, it is reasonable to look also for a map in the other direction. Such a map can be defined by associating with each object in the abstract model an object in the concrete model by a "free" construction, leading to a canonical representative in the concrete model for all objects which have the same abstract representation. If the dynamic behaviour of systems is also preserved by these maps, then they can be extended to functors between the corresponding categories. Within category theory the fact that such a pair of a "forgetful" functor and a "free" functor "fit together nicely" is expressed by the universality of the constructions as given through the notion of an adjunction. In this way an adjunction between two categories is a formal way to express that one model is more abstract than another model.

Ideally, the functor to the abstract model gives a faithful description of the concrete model in the sense that applying first the "free" and then the "forgetful" construction to an abstract object yields the same object (up to isomorphism). An adjunction with this property is called a *co-reflection*.

In comparing different models of concurrency via adjunctions we follow the approach laid out by others. The same method has also been used in, e.g., [94, 96, 68, 63, 98, 60].

The categorical approach also has the advantage that it allows one to treat useful constructions within a model in a uniform way. For instance, the categorical product of two objects in a category corresponds in general to the parallel composition of the two systems, whereas their co-product often corresponds to a non-deterministic choice between these systems. Having an adjunction between two categories now means that such universal constructions can often be transported easily from one model to the other.

With respect to Petri nets, their underlying algebraic structure is in particular useful for defining morphisms between them. As mentioned above, a Petri net can be viewed as a 2-sorted algebra with three operations [95]. This leads to a natural

notion of morphisms as maps between the sorts which preserve the three operations. In these morphisms a transition in one Petri net is simulated by a multiset of transitions in another Petri net. However, in [95] it is argued that such a general notion of morphism is often undesirable, so that an extra restriction is imposed which requires that a transition in one Petri net is simulated by at most one transition in another Petri net. For 1-safe Petri nets the resulting morphisms lead to a category for which there exist adjunctions with the categories of occurrence nets and prime event structures [94].

In [68] it appeared to be necessary to use a slight modification of these morphisms in order to obtain a co-reflection between the category of elementary net systems and the category of elementary transition systems. Similar modifications have been used in [63] in the context of Petri nets in order to obtain a co-reflection between a category of transition systems and the category of Petri nets. In this thesis we also use these Petri net morphisms.

A notion of Petri net morphism which is similar to the one from [95] can also be obtained via the graph representation of Petri nets [59]. These morphisms from [59] are defined as graph morphisms which are in addition monoid morphisms when restricted to the nodes. For 1-safe Petri nets the two notions of morphism coincide [60].

0.4 Historical Background

In this section we give some background concerning the history of net based models.

The foundations for net theory were laid by C.A Petri in 1962 [74, 75]. In the following years the research concentrated mainly on the model of C/E systems, see, e.g., [38].

The investigation of the non-sequential behaviour of net based models has been carried out mostly in the context of these C/E systems or variants such as elementary net systems. This investigation started with the introduction of processes of C/E systems by Petri [76]. Since then the underlying causal nets were also extensively investigated in their own right, see, e.g., [12, 27, 11].

In [55] Mazurkiewicz' traces were introduced in order to represent the non-sequential runs of C/E systems. Since then trace theory has developed into a model for concurrency which has an extensive theory on its own. See [20] for an overview. Research directions within trace theory include the connection with the existing theory of free partially commutative monoids [15, 8], the representation of traces as dependence graphs [2], and the study of infinite traces [29].

Prime event structures were introduced by Nielsen, Plotkin, and Winskel in [66] as concrete representations of certain Scott domains. This paper has also established the relationship of prime event structures to (a variant of) C/E systems. The connection between prime event structures and Scott's information systems [85] also led to the investigation of several generalizations of prime event structures by Winskel [92, 96].

The investigation of the model of Petri nets used in this thesis started in the early 70's. An early reference in which these systems appear is [37].

The early research on Petri nets took mainly the point of view of classical automata theory and dealt with issues concerning decidability, complexity, and formal languages. Several basic decidability results were given by Karp and Miller in [49] for Vector Addition Systems, a model equivalent to Petri nets. Their results include, e.g., the decidability of boundedness, i.e. the problem of existence of an upper bound on the number of tokens that any place can have in a reachable marking. One of the most famous results in this area is the decidability of the reachability problem, i.e. the problem whether a given marking is reachable [54, 50]. Other important results include the undecidability of marking equivalence, i.e. the problem whether two Petri nets have the same set of reachable markings [36]. An overview of work on decidability and complexity issues for Petri nets is given in [26]. Classes of formal languages generated by Petri nets were investigated among others by Hack [36] and Peterson [72, 73]. An overview of the developments in this area is given in [45]. Whereas the above mentioned work concentrates on the sequential languages generated by Petri nets, the non-sequential languages generated by Petri nets, called subset languages, have been investigated in [82].

On the one hand, as described above, net based models have been given a semantics in terms of other models. On the other hand, there has also been research on how nets themselves can be used in order to give a "true concurrency" semantics to other models, see, e.g., [93, 17, 32, 71, 25]. One of the advantages of this line of research has been a better structural understanding of Petri nets. Also the research on refinement operations (see, e.g., [69]), categorical constructions [95, 63], and the Petri Box calculus [10] has led to a better insight into the structure of Petri nets.

Whereas for modelling practical systems Petri nets are more attractive than C/E systems or elementary net systems, their applicability in practical situations is still limited. To overcome this problem many generalizations of Petri nets have been studied. These include timed Petri nets, stochastic Petri nets, Petri nets with inhibitor arcs, high-level nets, etc. One of the most successful generalizations of Petri nets is the generalization to Coloured Petri nets developed by Jensen [46]. These Coloured Petri nets are streamlined variants of the high-level nets introduced by Genrich and Lautenbach [30]. The initial motivation for the introduction of Coloured Petri nets was to allow an easier computation of invariants. The model of Coloured Petri nets extends the model of ordinary Petri nets by adding structure to the tokens in places. Later they were also made hierarchical in order to extend their applicability. An overview of the theory of Coloured Petri nets and of some of their industrial applications is given in [47].

0.5 Outline of the Thesis

After the preliminaries given in Chapter 1, Petri nets are introduced in Chapter 2. In this chapter also morphisms between Petri nets are defined which leads to the category of Petri nets.

In Chapter 3 multiset transition systems are introduced. The notion of a region, which plays a central role in this thesis, is defined in terms of these multiset transition

systems. Then, based on results from Mukund [63, 64], the semantics for Petri nets in terms of multiset transition systems is investigated. Characterizations are given of the classes of multiset transition systems associated with Petri nets and 1-safe Petri nets. Finally, the universality of the constructions is stated in categorical terms.

In Chapter 4 local traces are introduced in order to give a trace semantics for Petri nets. A characterization is given of the class of local trace languages associated with Petri nets. Based on this characterization the universality of the construction is proved. Then the trace semantics is compared with the classical trace semantics for 1-safe Petri nets in terms of Mazurkiewicz' traces. Finally in this chapter, the relationship to the approach from [9] is investigated.

Chapter 5 introduces local event structures in order to give an event structure semantics for Petri nets. The semantics for Petri nets in terms of a subclass of these local event structures is defined and it is proved that it extends the classical semantics for 1-safe Petri nets in terms of prime event structures. Then the universality of the semantics is proved for the subcategory of Petri nets without auto-concurrency. At the end of this chapter a possible extension of local event structures that allows to deal with auto-concurrency is discussed.

In Chapter 6 the relationship between local event structures introduced in Chapter 5 and some well-known classes of event structures is investigated in categorical terms. Finally, the Discussion in Chapter 7 mentions some open problems with respect to the previous chapters that we consider worth investigating.

Chapter 1

Preliminaries

In this chapter we fix notations and conventions used throughout the thesis.

1.1 Multisets, Sequences, Functions, and Partial Orders

Let A be a set, possibly empty or infinite. The set of finite subsets of A is denoted by $P_F(A)$. Elements of $P_F(A)$ will be referred to as *steps* (*over* A).

A *multiset* (*over* A) is a function $u : A \to \mathbf{N}$. A multiset u over A is *finite* if $\sum_{a \in A} u(a) < \infty$. The set of finite multisets over A is denoted by $M_F(A)$. We use u, u', u_1, u_2, v, etc. to range over $M_F(A)$. Note that the empty multiset $\underline{0} : A \to \mathbf{N}$ with $\underline{0}(a) = 0$ for all $a \in A$, is a member of $M_F(A)$.

For $u \in M_F(A)$, $|u| = \sum_{a \in A} u(a)$ is the number of elements in u. For two multisets $u, v \in M_F(A)$, their sum $u + v \in M_F(A)$ is defined by $(u + v)(a) = u(a) + v(a)$ for all $a \in A$; we write $v \leq u$ if $u = v + w$ for some $w \in M_F(A)$. If $u, v \in M_F(A)$ are such that $v \leq u$, then v is called a *submultiset* of u and $u - v$ is the (unique) multiset w such that $u = v + w$. The sum of an arbitrary finite set of finite multisets $\{u_i \in M_F(A) \mid i \in I\}$ is denoted by $\sum_{i \in I} u_i$. For $a \in A$ and $k \in \mathbf{N}$ we let $k \cdot a$ denote the finite multiset over A with $(k \cdot a)(b) = k$ if $b = a$ and $(k \cdot a)(b) = 0$ otherwise.

By A^* we denote the free monoid generated by A. The product operation is concatenation and the elements of A^* are called *sequences* (*over* A). The unit element of A^* is the empty word Λ. Let $A^+ = A^* - \{\Lambda\}$ be the set of non-empty sequences over A.

Elements of $(P_F(A))^+$ will be referred to as *step sequences* (*over* A). We view $(P_F(A))^+$ as a (free) monoid: the unit element is $\emptyset \in P_F(A)$ and the product operation is the accordingly modified usual concatenation operation. Thus $\rho\emptyset = \emptyset\rho = \rho$ for all $\rho \in (P_F(A))^+$, where $\rho\emptyset$ denotes the product of ρ and $\emptyset$.

Elements of $(M_F(A))^+$ will be referred to as *multiset sequences* (*over* A). We also view $(M_F(A))^+$ as a (free) monoid; the unit element is $\underline{0} \in M_F(A)$ and the product operation is the accordingly modified usual concatenation operation. Thus $\rho\underline{0} = \underline{0}\rho = \rho$.

In the context of sequences we may refer to A as an *alphabet*.

In this thesis we view sequences, i.e. elements from A^*, as step sequences, i.e. elements from $(P_F(A))^+$. Similarly, we view step sequences as multiset sequences. Formally this can be achieved by defining monoid homomorphisms $s_A : A^* \to (P_F(A))^+$ and $m_A : (P_F(A))^+ \to (M_F(A))^+$ in the following way: $s_A(\Lambda) = \emptyset$, $s_A(a) = \{a\}$, for all $a \in A$, and $m_A(\emptyset) = \underline{0}$, $m_A(u) \in M_F(A)$ with $u \in P_F(A)$ is given by $m_A(u)(a) = 1$ if $a \in u$ and $m_A(u)(a) = 0$ otherwise. Throughout the thesis we will avoid this notational complication and, e.g., simply write a for $s_A(a)$ and u for $m_A(u)$. All notions defined for $(M_F(A))^+$ are carried over to $(P_F(A))^+$ and A^* through the maps s_A and m_A.

For $a \in A$ and $\rho \in (M_F(A))^+$, we let $num_a(\rho)$ denote the number of times a occurs in ρ. Thus

$$num_a(\underline{0}) = 0 \text{ and } num_a(\rho u) = num_a(\rho) + u(a).$$

For $\rho \in (M_F(A))^+$, let $mset(\rho)$ denote the finite multiset (over A) of elements of A in ρ. Thus for all $a \in A$,

$$mset(\rho)(a) = num_a(\rho).$$

For $\rho \in (M_F(A))^+$, let $|\rho|$ denote the number of elements in ρ, that is

$$|\rho| = |mset(\rho)|;$$

we let $alph(\rho)$ denote the set of elements of A occurring in ρ, that is

$$alph(\rho) = \{a \in A \mid num_a(\rho) > 0\}.$$

Finally, for $A' \subseteq A$, let $proj_{A'} : (M_F(A))^+ \to (M_F(A'))^+$ be the homomorphism which erases all symbols which are not in A'. Thus, for $u \in M_F(A)$, $proj_{A'}(u) \in M_F(A')$ is given by $proj_{A'}(u)(a) = u(a)$ for all $a \in A'$.

Let A and B be sets and let $f : A \to B$ be a partial function. Then for $u \subseteq A$ and $v \subseteq B$, let

$$f(u) = \{b \in B \mid \exists a \in u.\, f(a) = b\}$$

and let

$$f^{-1}(v) = \{a \in A \mid \exists b \in v.\, f(a) = b\}.$$

The function f is lifted to multisets in the following way. Let $\hat{f} : M_F(A) \to M_F(B)$ be the *multiset extension* of f, given by:

$$\hat{f}(u)(b) = \sum_{f(a)=b} u(a).$$

Thus $\hat{f}(u)(b) = 0$ if $b \notin f(A)$. Note that if u is a step over A, then the multiset $\hat{f}(u)$ is in general not a step over B due to the fact that f may not be injective on u, so that $f(u) \neq \hat{f}(u)$. The homomorphic extension of $\hat{f}$ to multiset sequences is also denoted by $\hat{f}$. By viewing ordinary sequences as step sequences, and step sequences as multiset sequences, this also defines $\hat{f}$ on ordinary sequences and step sequences. To simplify the notation, we write f rather than $\hat{f}$ in what follows. Because we do not consider multiset extensions with domain $P_F(A)$ this will not lead to ambiguities.

Given a partial order $(X, \leq)$, we let for $x \in X$,

$$\downarrow x = \{y \in X \mid y \leq x\}$$

be the *downward-closure* of x (with respect to $\leq$). For $x, y \in X$ we say that x and y are *compatible* (under $\leq$), denoted by $x \uparrow y$, if

$$\exists z \in X.(x \leq z \text{ and } y \leq z).$$

1.2 Category Theory

In the Introduction in Chapter 0 it has been argued that category theory is a suitable framework for comparing different models. In this section we briefly mention some basic notions and results from category theory which are used in this thesis. We do not give any formal definitions. These can be found in, e.g., [53, 5, 77].

A *category* consists of a collection of *objects* and a collection of *morphisms* between objects which contains for each object A an *identity morphism* id_A and which has a *composition operator* "$\circ$" for morphisms.

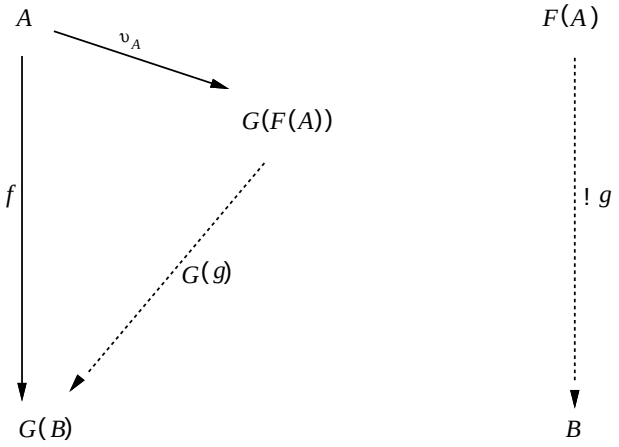
A notion of morphism induces in a standard way a notion of *isomorphism*: a morphism $f : A \to B$ is called an isomorphism if there exists a morphism $g : B \to A$ such that $g \circ f = id_A$ and $f \circ g = id_B$. In that case A and B are said to be *isomorphic*, and this is denoted by $A \equiv B$.

Restricting the objects and morphisms of a category, while preserving identity and composition, leads to the notion of a *subcategory*. If for every two objects in the subcategory *all* morphisms in the original category between these objects are also morphisms in the subcategory, then the subcategory is *full*. If the subcategory has the same objects as the original category, then the subcategory is *wide*.

Categories can be related by *functors*. A functor F from a category $\mathcal{C}$ to a category $\mathcal{D}$ is a map from the objects and morphisms of $\mathcal{C}$ to the objects and morphisms of $\mathcal{D}$ respectively, such that a morphism f from A to B is mapped to a morphism $F(f)$ from $F(A)$ to $F(B)$, identities are preserved, and compositions are preserved. The functor F is called *full* if for every two objects A and B of $\mathcal{C}$ and for every morphism g from $F(A)$ to $F(B)$ there exists a morphism f from A to B such that $F(f) = g$. The functor F is called *faithful* if for every two objects A and B of $\mathcal{C}$ and for every two morphisms f and g from A to B, $F(f) = F(g)$ implies that $f = g$.

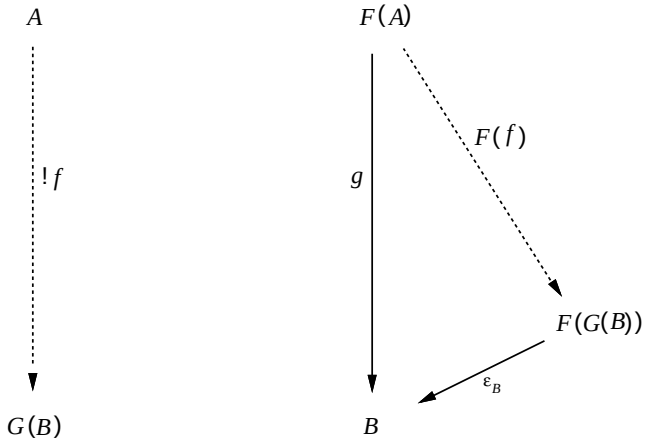
If $\mathcal{C}$ is a subcategory of $\mathcal{D}$ then $i_{\mathcal{C},\mathcal{D}}$, or briefly i, denotes the *inclusion functor* from $\mathcal{C}$ to $\mathcal{D}$. Then for an object A of $\mathcal{C}$ and a morphism f of $\mathcal{C}$, we will often simply write A and f for the object $i(A)$ of $\mathcal{D}$ and the morphism $i(f)$ of $\mathcal{D}$, respectively.

The most important notion from category theory in this thesis is the notion of an *adjunction*. An adjunction between two categories $\mathcal{C}$ and $\mathcal{D}$ consists of a functor F from $\mathcal{C}$ to $\mathcal{D}$, a functor G from $\mathcal{D}$ to $\mathcal{C}$, and morphisms v_A from A to $G(F(A))$ for every object A of $\mathcal{C}$, with the following property. Suppose A is an object of $\mathcal{C}$, B is an object of $\mathcal{D}$, and f is a morphism from A to $G(B)$. Then there exists a unique morphism g from $F(A)$ to B such that the following diagram commutes.



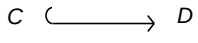
The morphisms υ_A form the *unit* of the adjunction. The functor F is called the *left adjoint* and the functor G the *right adjoint*.

Alternatively, F and G can be proved to be an adjunction by showing the existence of morphisms ϵ_B from $F(G(B))$ to B for every object B of $\mathcal{D}$, with the following property. If A is an object of $\mathcal{C}$, B is an object of $\mathcal{D}$, and g is a morphism from $F(A)$ to B, then there exists a unique morphism f from A to $G(B)$ such that the following diagram commutes.

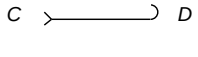


The morphisms ϵ_B form the *co-unit* of the adjunction.

If the morphisms which form the unit of the adjunction are isomorphisms, then the adjunction is called a *co-reflection*. In this case we write



to express that there is a co-reflection between $\mathcal{C}$ and $\mathcal{D}$. If the morphisms which form the co-unit of the adjunction are isomorphisms, then the adjunction is called a *reflection*. In this case



is written to express the fact that there is a reflection between $\mathcal{C}$ and $\mathcal{D}$. If both the unit and co-unit consist of isomorphisms, then the categories $\mathcal{C}$ and $\mathcal{D}$ are *(categorically) equivalent*. A logically equivalent characterization is that $\mathcal{C}$ and $\mathcal{D}$ are equivalent iff F is a full and faithful functor, and for every object B of $\mathcal{D}$ there exists an object A of $\mathcal{C}$ such that $F(A) \equiv B$.

Adjunctions can be composed in the following way. Suppose $F : \mathcal{C} \to \mathcal{D}$ and $G : \mathcal{D} \to \mathcal{C}$ form an adjunction with F the left adjoint, and $F' : \mathcal{D} \to \mathcal{E}$ and $G' : \mathcal{E} \to \mathcal{D}$ form an adjunction with F' the left adjoint. Then the compositions $F' \circ F : \mathcal{C} \to \mathcal{E}$ and $G \circ G' : \mathcal{E} \to \mathcal{C}$ also form an adjunction, with $F' \circ F$ the left adjoint. Moreover, if both adjunctions are co-reflections then also their composition is a co-reflection. Similarly, if both adjunctions are reflections, then their composition is also a reflection.

Chapter 2

Petri Nets

In this chapter the model of Petri nets is introduced. In Section 2.1 the Petri nets which form the main subject of investigation are defined. We also introduce in this section the subclasses of 1-safe Petri nets and co-safe Petri nets. Some observations on behavioural aspects of the various classes are formulated. Finally, in Section 2.2 we equip Petri nets with structure-preserving morphisms, leading to a category of Petri nets.

2.1 Petri Nets

A Petri net has as its underlying structure a weighted directed bipartite graph. The nodes are partitioned into *places* and *transitions*. Places are *marked*, i.e. they contain an arbitrary finite number of *tokens*. A marked place is a local state of the Petri net. The transitions represent the actions of the system. The arc-weights are given by a *weight function*. This weight function gives the number of tokens each transition takes from each place and puts in each place if it occurs. The initial global state of the Petri net is an *initial marking* of the places, defined as a multiset of places.

Definition 2.1.1
A *Petri net* is a quadruple $N = (S, T, W, M_{in})$ where

- S is a set of *places* and T is a set of *transitions* such that $S \cap T = \emptyset$
- $W : (S \times T) \cup (T \times S) \to \mathbf{N}$ is a *weight function*
- $M_{in} : S \to \mathbf{N}$ is the *initial marking* of N. □

Note that a Petri net may be infinite. Also note that a Petri net can have isolated elements.

Let $\mathcal{M}_N$ denote the set of all *markings* of a Petri net N, i.e. the set of all functions $M : S \to \mathbf{N}$ where S is the set of places of N.

A Petri net is represented graphically by drawing its places as circles, its transitions as boxes, and by drawing a directed arc between a place and a transition if the corresponding weight is greater than 0. If this weight is greater than 1 then the arc is

labelled with this weight. The initial marking is represented by drawing in each place its number of tokens.

In Figure 2.1 and Figure 2.2 the Petri nets N_1 and N_2 are depicted which will be used frequently as an example.

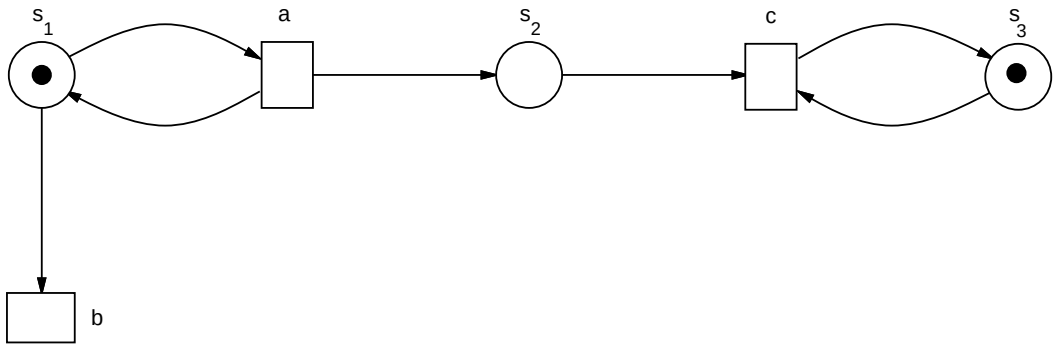


Figure 2.1: The Petri net N_1

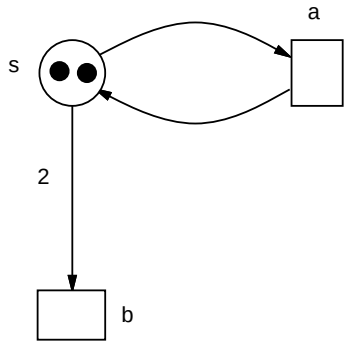


Figure 2.2: The Petri net N_2

Given a Petri net $N = (S, T, W, M_{in})$ and an element $x \in S \cup T$, let

$$({}^\bullet x)_N = \{y \in S \cup T \mid W(y, x) > 0\}$$

be the set of *input* elements of x (in N) and let

$$(x^\bullet)_N = \{y \in S \cup T \mid W(x, y) > 0\}$$

be the set of *output* elements of x (in N).

The dynamic behaviour of Petri nets is defined by a firing rule. This firing rule describes when (finite multisets of) transitions can occur at a marking, and what the effect is when they do.

Definition 2.1.2
Let $N = (S, T, W, M_{in})$ be a Petri net, let $M, M' \in \mathcal{M}_N$, and let $u \in M_F(T)$. Then

(1) u is *enabled at* M, denoted by $M[u\rangle_N$, if

$$\forall s \in S.\, M(s) \geq \sum_{t \in T} u(t) \cdot W(s,t)$$

(2) u *can occur at* M and *lead to* M', denoted by $M[u\rangle_N M'$, if

$$M[u\rangle_N \text{ and } \forall s \in S.\, M'(s) = M(s) + \sum_{t \in T} u(t) \cdot (W(t,s) - W(s,t)).$$

□

Thus a transition t is enabled at a marking M if each of its input places s has a sufficient number of tokens: $M(s) \geq W(s,t)$. An arbitrary finite multiset of transitions is enabled at a marking if each place contains enough tokens for each transition in the multiset.

Note that if N is a Petri net and $M[u\rangle_N M'$, then also $M[v\rangle_N$ for all $v \leq u$. Moreover, if $v \leq u$ and $M'' \in \mathcal{M}_N$ is such that $M[v\rangle_N M''$, then $M''[u - v\rangle_N M'$. This observation, that occurrences of multisets can be split arbitrarily, will be frequently used in the sequel.

The effect of an occurrence of a transition is purely local in that it only influences the markings of the places it is connected to. In particular, a step consisting of transitions with disjoint environments is enabled at a marking if the transitions are enabled individually.

Lemma 2.1.3
Let $N = (S, T, W, M_{in})$ be a Petri net, let $M \in \mathcal{M}_N$, and let $u \in P_F(T)$ be such that

$$(\forall t \in u.\, M[t\rangle_N) \text{ and}$$

$$\forall t_1, t_2 \in u.\, (t_1 \neq t_2 \Rightarrow ({}^\bullet t_1)_N \cup (t_1{}^\bullet)_N) \cap ({}^\bullet t_2)_N \cup (t_2{}^\bullet)_N) = \emptyset).$$

Then

$$M[u\rangle_N.$$

□

When analyzing the behaviour of Petri nets it is not necessary to consider all its markings; only the *reachable* markings are of interest.

Definition 2.1.4
Let $N = (S, T, W, M_{in})$ be a Petri net. The set $\mathcal{RM}_N \subseteq \mathcal{M}_N$ of *reachable markings* of N is the least set containing M_{in} such that

$$(M \in \mathcal{RM}_N \text{ and } M[u\rangle_N M') \Rightarrow M' \in \mathcal{RM}_N.$$

□

In defining the dynamic behaviour of Petri nets we use multisets of transitions rather than sets. The Petri net N_2 depicted in Figure 2.2 illustrates that it is indeed possible that multiple instances of a transition are enabled at a reachable marking, because at its initial marking there are enough tokens in s for two instances of the transition a to occur. This phenomenon is called *auto-concurrency*. As we will see in Chapter 5 auto-concurrency may severely complicate the behavioural analysis of Petri nets. In several simpler models of net theory such as 1-safe Petri nets, elementary net systems and safe net systems auto-concurrency cannot occur.

For analyzing the behaviour of Petri nets we are not just interested in single occurrences of multisets of transitions, but rather in sequences of these occurrences starting from the initial marking.

Definition 2.1.5
Let $N = (S, T, W, M_{in})$ be a Petri net.

(1) An *occurrence sequence* of N is a sequence $M_0 u_1 M_1 u_2 \dots u_n M_n$ with $n \geq 0$ and with $u_1, \dots, u_n \in M_F(T)$ and $M_0, \dots, M_n \in \mathcal{RM}_N$ such that

- $M_0 = M_{in}$
- $\forall 1 \leq i \leq n.\, M_{i-1}[u_i\rangle_N M_i$.

(2) The set $MFS_N \subseteq (M_F(T))^+$ of *multiset firing sequences* of N is the set of all $\rho \in (M_F(T))^+$ for which there exists an occurrence sequence $M_0 u_1 M_1 u_2 \dots u_n M_n$ of N with $\rho = u_1 \dots u_n$. □

Multiset firing sequences are more abstract than occurrence sequences in the sense that two Petri nets which have the same set of multiset firing sequences may have different sets of occurrence sequences. For a fixed Petri net however, each multiset firing sequence uniquely determines the intermediate markings.

Given a Petri net N and $\rho \in MFS_N$ with $\rho = u_1 \dots u_n$, let $(M_\rho)_N$ denote the unique marking M_n of N such that $M_0 u_1 M_1 u_2 \dots u_n M_n$ is an occurrence sequence of N.

Due to the possibility of auto-concurrency in Petri nets, we have associated with a Petri net a set of *multiset* sequences rather than a set of *step* sequences. In Chapter 5 it will sometimes be necessary to "ignore" auto-concurrency from the behaviour of Petri nets. This can be done through the following notion.

Definition 2.1.6
Let $N = (S, T, W, M_{in})$ be a Petri net. The set $SFS_N \subseteq (P_F(T))^+$ of *step firing sequences* of N is given by

$$SFS_N = MFS_N \cap (P_F(T))^+.$$

□

In the above definition we use the convention given in Chapter 1 that step sequences are viewed as multiset sequences.

For some of our constructions, it is for the categorical results necessary to restrict our attention to *co-safe* Petri nets, which do not exhibit any auto-concurrency at all in their behaviours.

Definition 2.1.7
A Petri net N is *co-safe* if $MFS_N = SFS_N$. □

The Petri net N_1 depicted in Figure 2.1 is co-safe, whereas the Petri net N_2 depicted in Figure 2.2 is not co-safe. Co-safe Petri nets arise for instance as the targets of the net semantics constructed for the process algebra called Petri Box Calculus [10]. This follows from the work of [19].

Another abstraction can be made from the set of multiset firing sequences by only considering the sequential representatives.

In the following definition we use the convention from Chapter 1 that ordinary sequences are viewed as step sequences by the monoid homomorphism which maps each element of the alphabet to the singleton containing this element.

Definition 2.1.8
Let $N = (S, T, W, M_{in})$ be a Petri net. The set $FS_N \subseteq T^*$ of *firing sequences* of N is given by

$$FS_N = SFS_N \cap T^*.$$

□

Now we turn to the important subclass of *1-safe* Petri nets. A 1-safe Petri net has the property that it has no reachable markings in which a place carries 2 or more tokens.

Note that in any Petri net with this property, transitions that are connected to a place by an arc with weight 2 or more, will never be enabled at a reachable marking. As a consequence, these transitions and the arcs incident with them can be removed from the Petri net without affecting its set of multiset firing sequences. Hence we might as well exclude such transitions right away in the definition of 1-safe Petri nets.

Usually when the behaviour of 1-safe Petri nets is investigated, as in, e.g., [96, 9, 24, 60, 64], a restriction is imposed which forbids isolated transitions. For 1-safe Petri nets it is this condition which prevents auto-concurrency. Also for our results on 1-safe Petri nets we exclude auto-concurrency by adopting such a restriction.

The above considerations lead to the following formal definition of 1-safe Petri nets.

Definition 2.1.9
A Petri net $N = (S, T, W, M_{in})$ is *1-safe* if

(1) $\forall M \in \mathcal{RM}_N. \forall s \in S. M(s) \leq 1$

(2) $\forall s \in S. \forall t \in T. (W(s,t) \in \{0,1\} \text{ and } W(t,s) \in \{0,1\})$

(3) $\forall t \in T.\, ({}^\bullet t)_N \cup (t^\bullet)_N \neq \emptyset$. □

Observe that (1) and (3) together imply that every transition has at least one input place. Also observe that, while (2) and (3) are conditions imposed on the structure of the Petri net, the "crucial" characteristic property (1) is not a structural property but defined in behavioural terms, i.e. reachable markings.

In Figure 2.3 and Figure 2.4 two 1-safe Petri nets N_3 and N_4 are depicted. These Petri nets will be used to illustrate the several behavioural notions for 1-safe Petri nets.

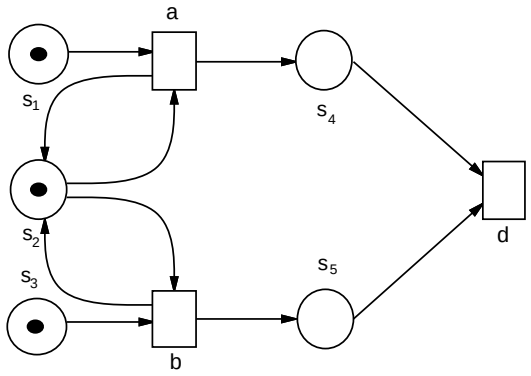


Figure 2.3: The 1-safe Petri net N_3

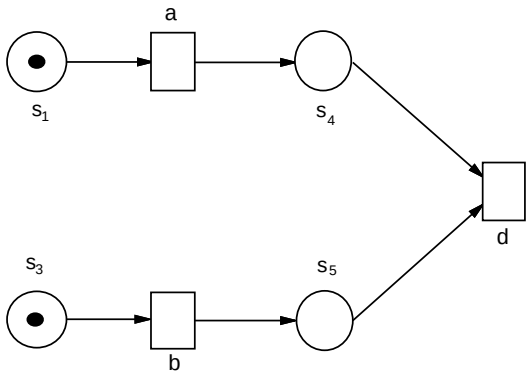


Figure 2.4: The 1-safe Petri net N_4

Since 1-safe Petri nets do not exhibit auto-concurrency, every 1-safe Petri net is co-safe. The class of co-safe Petri nets is however a non-trivial extension of the class of 1-safe Petri nets. The Petri net N_1 depicted in Figure 2.1 is an example of a Petri net which is co-safe, but not 1-safe.

By Lemma 2.1.3 a finite set of transitions of a Petri net is enabled at a reachable marking if the transitions are enabled individually and have disjoint environments. From the following easy to prove lemma it follows that for 1-safe Petri nets the converse also holds.

Lemma 2.1.10

Let $N = (S, T, W, M_{in})$ be a 1-safe Petri net, let $M \in \mathcal{RM}_N$, and let $u \in P_F(T)$. Then

$$M[u\rangle_N \Leftrightarrow$$

$$((\forall t \in u.\, M[t\rangle_N) \text{ and}$$

$$\forall t_1, t_2 \in u.\, (t_1 \neq t_2 \Rightarrow ((^\bullet t_1)_N \cup (t_1{}^\bullet)_N) \cap ((^\bullet t_2)_N \cup (t_2{}^\bullet)_N) = \emptyset)).$$

□

Thus for a 1-safe Petri net the concurrent behaviour in terms of its step firing sequences can be recovered from its sequential behaviour by the binary relation over its transitions consisting of all pairs of transitions with disjoint environments.

Another consequence of Lemma 2.1.10 is that concurrent steps within 1-safe Petri nets can be recovered from concurrent pairs of transitions.

Lemma 2.1.11

Let $N = (S, T, W, M_{in})$ be a 1-safe Petri net, let $M \in \mathcal{RM}_N$, and let $u \in P_F(T)$. Then

$$M[u\rangle_N \Leftrightarrow \forall t_1, t_2 \in u.\, M[\{t_1, t_2\}\rangle_N.$$

□

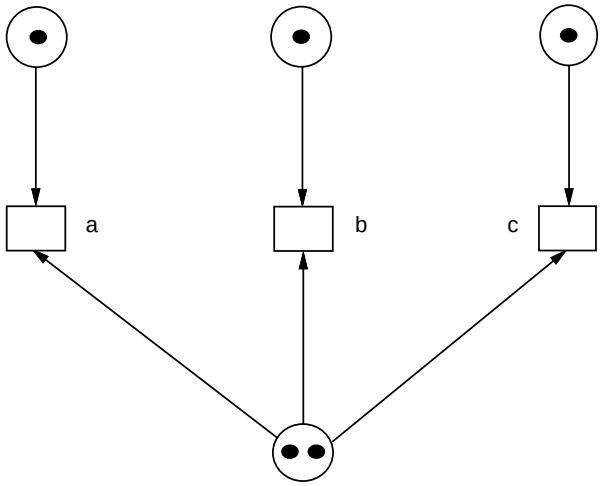


Figure 2.5: The Petri net N_5

For a general Petri net it is possible that a set of transitions is enabled at one reachable marking, but not at another reachable marking, even though at both markings the transitions are enabled individually. This is for instance the case for the Petri net N_5 depicted in Figure 2.5. In this Petri net the set $\{a, b\}$ is enabled initially, but if c occurs first then only one of the transitions a and b can occur. As a consequence of Lemma 2.1.10 such a situation cannot arise for 1-safe Petri nets. For 1-safe Petri nets it is possible to characterize concurrency (and conflict) as a *global* property.

Lemma 2.1.12

Let $N = (S, T, W, M_{in})$ be a 1-safe Petri net, let $M \in \mathcal{RM}_N$, and let $u \in P_F(T)$. Then

$$M[u\rangle_N \Leftrightarrow ((\forall t \in u.\, M[t\rangle_N) \text{ and } \exists M' \in \mathcal{RM}_N.\, M'[u\rangle_N).$$

□

Several of the notions we consider in the context of Petri nets are generalizations of corresponding notions defined first in the context of net classes closely related to 1-safe Petri nets such as *elementary net systems* and *safe net systems*, which are variants of the original model of C/E systems. To conclude this section we briefly introduce these models. The purpose is to argue that it is indeed possible to transport several behavioural notions defined originally for these net systems to the context of (1-safe) Petri nets. In the rest of this thesis these net systems will however only play a minor role.

For elementary net systems, see, e.g., [81], we speak of *conditions* instead of places, *events* instead of transitions, and *cases* instead of markings.

Definition 2.1.13

An *elementary net system* is a quadruple $N = (B, E, F, c_{in})$ where

(1) B is a set of *conditions* and E is a set of *events* such that $B \cap E = \emptyset$

(2) $F \subseteq (B \times E) \cup (E \times B)$ is a *flow relation* such that

$$\forall e \in E.\, \exists b \in B.\, ((b, e) \in F \text{ or } (e, b) \in F).$$

(3) $c_{in} \subseteq B$ is the *initial case* of N. □

Given an elementary net system $N = (B, E, F, c_{in})$, let for $x \in B \cup E$,

$$({}^\bullet x)_N = \{y \in B \cup E \mid (y, x) \in F\}$$

be the set of *input* elements of x (in N) and let

$$(x^\bullet)_N = \{y \in B \cup E \mid (x, y) \in F\}$$

be the set of *output* elements of x (in N).

Sometimes it is demanded that the underlying net of an elementary net system is *simple*, which means that every element is uniquely characterized by its input and

output elements. Following [64] this is not required here. Usually it is also demanded that an elementary net system has no isolated conditions. With respect to the notions considered in this thesis, this restriction is however not necessary. Hence we have omitted it, in order to streamline the definition with the definition of 1-safe Petri nets.

The global states of an elementary net system are given by sets of conditions which hold. For an elementary net system $N = (B, E, F, c_{in})$, let $\mathcal{C}_N$ denote the set of all these global states, which are called *cases*. Thus $\mathcal{C}_N$ is the set of all subsets of conditions of N.

The class of 1-safe Petri nets is defined in terms of reachable markings. The class of elementary net systems on the other hand has a purely *structural* definition. For elementary net systems "1-safeness" is ensured by the definition of its firing rule.

For an elementary net system an event can occur at a given case if all its input conditions hold and none of its output conditions hold. The effect of the occurrence of an event is that all its input conditions cease to hold and all its output conditions begin to hold.

Definition 2.1.14

Let $N = (B, E, F, c_{in})$ be an elementary net system, let $c, c' \in \mathcal{C}_N$, and let $e \in E$. Then

(1) e is *enabled at* c, denoted by $c[e\rangle_N$, if

$$^\bullet e \subseteq c \text{ and } e^\bullet \cap c = \emptyset$$

(2) e *can occur at* c and *lead to* c', denoted by $c[e\rangle_N c'$, if

$$c[e\rangle_N \text{ and } c' = (c - {}^\bullet e) \cup e^\bullet.$$

□

Note that for elementary net systems concurrency is not represented explicitly in the definition of the dynamic behaviour. The reason is, that for elementary net systems concurrency can be recovered from the sequential behaviour through certain diamond properties [81, 39].

The set of *reachable* cases of an elementary net system is defined similar to the set of reachable markings of a Petri net.

Definition 2.1.15

Let $N = (B, E, F, c_{in})$ be an elementary net system. Then the set $\mathcal{RC}_N \subseteq \mathcal{C}_N$ of *reachable cases* of N is the least set containing c_{in} such that

$$(c \in \mathcal{RC}_N \text{ and } c[e\rangle_N c') \Rightarrow c' \in \mathcal{RC}_N.$$

□

In order to see if an event can occur both its input conditions and its output conditions must be inspected. For *contact-free* elementary net systems it is sufficient to inspect only the input conditions of events at reachable cases.

Definition 2.1.16

An elementary net system $N = (B, E, F, c_i)$ is *contact-free* if

$$\forall e \in E. \forall c \in \mathcal{RC}_N. ({}^\bullet e \subseteq c \Rightarrow e^\bullet \cap c = \emptyset).$$

□

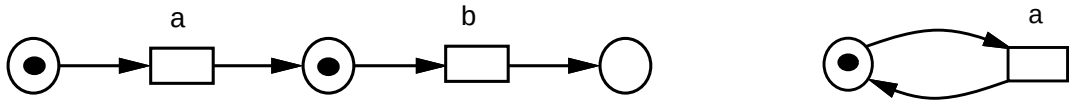


Figure 2.6: Two elementary net systems which are not contact-free

In Figure 2.6 two elementary net systems are depicted. Neither of them is contact-free. For the first elementary net system the event a can only occur if the event b occurs first. Whereas events are allowed to have self-loops, i.e. input and output conditions with a non-empty intersection as in the second elementary net system in Figure 2.6, these events can never occur.

As far as the behavioural notions in this thesis are concerned, there exists for each elementary net system a contact-free elementary net system which has the same behaviour. This contact-free elementary net system is obtained by complementing the conditions, see, e.g., [81]. It is easy to see that every contact-free elementary net system can also be viewed as a 1-safe Petri net which has the same (sequential) dynamic behaviour. On the other hand, viewing a 1-safe Petri net as an elementary net system does not always lead to a system which has the same dynamic behaviour, due to the difference in the firing rule in the presence of self-loops.

The above mentioned mismatch between contact-free elementary net systems and 1-safe Petri nets disappears for the variant of elementary net systems as used in [57, 98]. These net systems are here called *safe net systems*.

The underlying structure of safe net systems is the same as for elementary net systems. Thus a safe net system N is also a quadruple consisting of a set B of conditions, a set E of events, a flow relation F, and an initial case c_{in}. Given an element $x \in B \cup E$, its input elements $({}^\bullet x)_N$ and output elements $(x^\bullet)_N$ are defined in the same way as for elementary net systems. Again $\mathcal{C}_N$ denotes the set of all cases of N, which is the set of all subsets of conditions of N.

The only difference with elementary net systems is that safe net systems have a less restrictive firing rule.

Definition 2.1.17

Let $N = (B, E, F, c_{in})$ be a safe net system, let $c, c' \in \mathcal{C}_N$, and let $e \in E$. Then

(1) e is *enabled at* c, denoted by $c[e)_N$, if

$${}^\bullet e \subseteq c \text{ and } e^\bullet \cap (c - {}^\bullet e) = \emptyset$$

(2) e *can occur at* c and *lead to* c', denoted by $c[e\rangle_N c'$, if

$$c[e)_N \text{ and } c' = (c - {}^\bullet e) \cup e^\bullet.$$

□

Thus for a safe net system an event can occur if its input conditions hold and its output conditions only hold if these conditions are also input conditions of the event. The effect of the occurrence of an event is the same as for elementary net systems.

Definition 2.1.18

Let $N = (B, E, F, c_{in})$ be a safe net system. Then the set $\mathcal{RC}_N \subseteq \mathcal{C}_N$ of *reachable cases* of N is the least set containing c_{in} such that

$$(c \in \mathcal{RC}_N \text{ and } c[e)_N c') \Rightarrow c' \in \mathcal{RC}_N.$$

□

In view of the modifications in the firing rule for safe net systems with respect to the firing rule for elementary net systems, the definition of contact-freeness of safe net systems is modified accordingly.

Definition 2.1.19

A safe net system $N = (B, E, F, c_i)$ is *contact-free* if

$$\forall e \in E. \forall c \in \mathcal{RC}_N. ({}^\bullet e \subseteq c \Rightarrow e^\bullet \cap (c - {}^\bullet e) = \emptyset).$$

□

If we view the elementary net systems from Figure 2.6 as safe net systems, then the first safe net system is not contact-free, but the second safe net system is contact-free. In the first safe net system the event a can still not occur initially, but in the second system the event a can occur.

Whereas the class of contact-free elementary net systems can be viewed as a proper subclass of the class of 1-safe Petri nets, the classes of contact-free safe net systems and 1-safe Petri nets are essentially the same. Stated formally, the map obtained by viewing a contact-free safe net system as a 1-safe Petri net is a bijection between the two classes.

In the rest of this thesis we will often omit the subscript $_N$ for the notions defined in this section when the Petri net, elementary net system, or safe net system N is clear from the context.

2.2 The Category $\mathcal{PN}$

In this section we define morphisms between Petri nets which leads to a category of Petri nets.

Several proposals for Petri net morphisms have been made in the literature. Here we use the modified version of Winskel's morphisms from [95] which is used by Mukund in [63]. In [95] a Petri net morphism from N_1 to N_2 consists of a partial function from the transitions of N_1 to the transitions of N_2 and a multirelation from the places of N_1 to the places of N_2. These maps are required to preserve the initial marking and the environment of transitions.

In [63] it appeared to be necessary to modify these morphisms in order to show the universality of the transition system semantics for Petri nets. The modifications are twofold. Firstly the multirelation between the places is now required to be a partial function in the "reverse" direction rather than an arbitrary multirelation. Secondly the condition that the initial marking should be preserved is relaxed by only demanding preservation for "related" places.

These modifications with respect to Winskel's morphisms are similar to the modifications proposed in [68] in the context of elementary net systems. Recall that every elementary net system can be viewed as a Petri net, which is 1-safe if the elementary net system is contact-free. If we view elementary net systems as Petri nets in this way, the morphisms from [68] coincide with our notion of morphism between Petri nets.

Definition 2.2.1

Let $N_i = (S_i, T_i, W_i, M_i)$, $i = 1, 2$, be a pair of Petri nets. A *PN-morphism* from N_1 to N_2 is a pair (β, η) of partial functions $\beta : S_2 \to S_1$ and $\eta : T_1 \to T_2$ such that

(1) $\forall s_2 \in S_2.\, (\beta(s_2) \text{ is defined } \Rightarrow M_2(s_2) = M_1(\beta(s_2)))$

(2) $\forall t_1 \in T_1.\, (\eta(t_1) \text{ is undefined } \Rightarrow \beta^{-1}({}^\bullet t_1) = \beta^{-1}(t_1{}^\bullet) = \emptyset)$

(3) $\forall t_1 \in T_1.\, (\eta(t_1) \text{ is defined } \Rightarrow$

(3a) $\beta^{-1}({}^\bullet t_1) = {}^\bullet\eta(t_1)$ and $\beta^{-1}(t_1{}^\bullet) = \eta(t_1)^\bullet$ and

(3b) $\forall s_2 \in {}^\bullet\eta(t_1).\, W_2(s_2, \eta(t_1)) = W_1(\beta(s_2), t_1)$ and

(3c) $\forall s_2 \in \eta(t_1)^\bullet.\, W_2(\eta(t_1), s_2) = W_1(t_1, \beta(s_2)))$. □

In Figure 2.7 an example of a PN-morphism (β, η) is given, where the functions β and η are as indicated by the dotted arrows.

If (β, η) is a PN-morphism from N to N' then a transition t in N is "simulated" by $\eta(t)$ in N' whenever $\eta(t)$ is defined; otherwise it is "suppressed" in N'. The relation β^{-1} does not relate places which differ with respect to their initial marking. Moreover, this relation preserves the environment of transitions, thus ensuring that for places in N' which prevent the occurrence of $\eta(t)$, there exist corresponding places in N which prevent the occurrence of t. Consequently PN-morphisms preserve the dynamic behaviour of Petri nets. This is stated in the next lemma which is originally from [63].

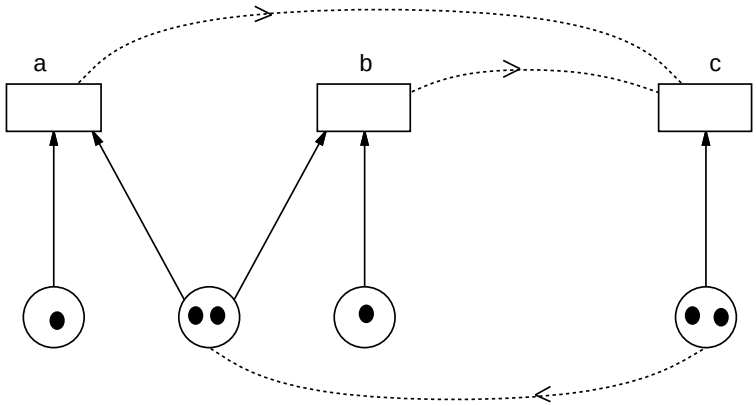


Figure 2.7: An example of a PN-morphism (β, η)

Lemma 2.2.2

Let $N_i = (S_i, T_i, W_i, M_i)$, $i = 1, 2$, be Petri nets and let (β, η) be a PN-morphism from N_1 to N_2. Define for $M \in \mathcal{RM}_{N_1}$, the marking $\hat{M} \in \mathcal{M}_{N_2}$ by:

$$\forall s \in S_2.\, \hat{M}(s) = \begin{cases} M(\beta(s)) & \text{if } \beta(s) \text{ is defined} \\ M_2(s) & \text{otherwise.} \end{cases}$$

If

$$M_0[u_1\rangle_{N_1} M_1[u_2\rangle_{N_1} \ldots [u_n\rangle_{N_1} M_n$$

is an occurrence sequence of N_1, then

$$\hat{M}_0[\eta(u_1)\rangle_{N_2} \hat{M}_1[\eta(u_2)\rangle_{N_2} \ldots [\eta(u_n)\rangle_{N_2} \hat{M}_n$$

is an occurrence sequence of N_2. □

Thus if (β, η) is a PN-morphism, then it preserves the dynamic behaviour in terms of multiset firing sequences (under η). PN-morphisms however also enforce restrictions on the structure of the Petri nets, which may not involve the dynamic behaviour. For instance, removing in the first Petri net in Figure 2.7 the place with two tokens in the initial marking does not change its dynamic behaviour in terms of multiset firing sequences, but it destroys the PN-morphism between the two Petri nets.

The Petri nets which arise out of our later constructions will all have the property that they are *S-simple*, which means that each place is uniquely characterized by its initial marking and its input and output transitions.

Definition 2.2.3

A Petri net (S, T, W, M_{in}) is *S-simple* if

$$\forall s_1, s_2 \in S.\, ((M_{in}(s_1) = M_{in}(s_2) \text{ and } \forall t \in T.\, (W(s_1, t) = W(s_2, t) \text{ and}$$
$$W(t, s_1) = W(t, s_2))) \Rightarrow s_1 = s_2).$$

□

For S-simple Petri nets without isolated places, a PN-morphism is completely determined by its transition function, which follows from another result by Mukund [63].

Lemma 2.2.4
Let (β_1, η) and (β_2, η) be a pair of PN-morphisms from N_1 to N_2 where N_1 is S-simple and for every place s of N_1, ${}^\bullet s \cup s^\bullet \neq \emptyset$. Then $\beta_1 = \beta_2$. □

With the notion of PN-morphism we can now define a category of Petri nets.

Definition 2.2.5
$\mathcal{PN}$ is the category which has Petri nets as its objects and PN-morphisms as its arrows. The identity morphism associated with an object is the pair of identity functions on places and transitions; composition of PN-morphisms (β_1, η_1) from N_1 to N_2 and (β_2, η_2) from N_2 to N_3 is the PN-morphism $(\beta_1 \circ \beta_2, \eta_2 \circ \eta_1)$ from N_1 to N_3. □

It is easy to verify that the composition of PN-morphisms is indeed a PN-morphism, so that the category $\mathcal{PN}$ is well-defined.

For the sublasses of Petri nets defined in Section 2.1 we have the following subcategories of $\mathcal{PN}$.

Definition 2.2.6
$\mathcal{PN}s$ is the full subcategory of $\mathcal{PN}$ the objects of which are 1-safe Petri nets.
$\mathcal{PNS}$ is the full subcategory of $\mathcal{PN}$ the objects of which are co-safe Petri nets. □

Note that $\mathcal{PN}s$ is also a full subcategory of $\mathcal{PNS}$.

With respect to universal constructions within the category of Petri nets, the empty net $(\emptyset, \emptyset, \emptyset, \underline{0})$ is both the initial and terminal object in $\mathcal{PN}$. As shown in [63], the category $\mathcal{PN}$ also has products and co-products. The product of two Petri nets corresponds to the synchronous parallel composition. The co-product of two Petri nets is however less standard. It can be viewed as either a non-deterministic choice or an asynchronous parallel composition, depending on the similarity of the two Petri nets. The morphisms considered in [95] and in [59] (for the category of Petri nets with an initial marking) give rise to categories which do not have co-products in general. There co-products are shown to exist for the subcategory of 1-safe Petri nets ([95]) and the category of Petri nets in which the initial marking is a set ([59]).

By Lemma 2.2.2, both multiset firing sequences and firing sequences are preserved under PN-morphisms. On the other hand, step firing sequences are in general not preserved under PN-morphisms due to the fact that PN-morphisms may map concurrent transitions in the first Petri net to the same transition in the second Petri net (as for the PN-morphism depicted in Figure 2.7). This leads to auto-concurrency in the second Petri net, so that *step* firing sequences in the first Petri net correspond to *multiset* firing sequences in the second Petri net.

This can be avoided by considering only PN-morphisms that map concurrent transitions to different transitions. Such PN-morphisms will be referred to as being *co-injective*.

Definition 2.2.7

Let (β, η) be a PN-morphism from $N_1 = (S_1, T_1, W_1, M_1)$ to N_2. Then (β, η) is *co-injective* if

$$\forall t, t' \in T_1 . ((t \neq t' \text{ and } \eta(t) \text{ and } \eta(t') \text{ are both defined and}$$

$$\exists M \in \mathcal{RM}_{N_1} . M[\{t, t'\}\rangle_{N_1}) \Rightarrow \eta(t) \neq \eta(t')).$$

□

The category of Petri nets with co-injective PN-morphisms will play a role in Section 5.7.

Definition 2.2.8

$\mathcal{PNC}$ is the wide subcategory of $\mathcal{PN}$ the arrows of which are co-injective PN-morphisms. □

Note that by Lemma 2.2.2 PN-morphisms between co-safe Petri nets are always co-injective. In other words, $\mathcal{PNS}$ is a full subcategory of $\mathcal{PNC}$.

Chapter 3

Multiset Transition Systems

With each Petri net we have associated a state space consisting of its reachable markings. These markings assign to each place of the Petri net a certain number of tokens and can thus be viewed as distributed global states, with the marked places as local states. For the analysis of the dynamic behaviour of a Petri net however, the exact distribution of the global states over local states is not essential.

Abstracting from this distribution leads to the model of *multiset transition systems* [52, 63], the subject of this chapter. The multiset transition system model is an easy to understand model for concurrent systems which is capable of capturing the concurrent and branching aspects of the behaviour of Petri nets, but which only has a notion of a global state.

Multiset transition systems are also a convenient framework for defining the basic notion of a *(generalized) region* [63, 40]. Regions provide a decomposition of global states into local states. The notion of a region plays a fundamental role in this thesis whenever we want to synthesize a Petri net from a behavioural description. Regions then yield places of Petri nets. The regions we use here form a generalization of the notion of a region from [22].

Thus the purpose of this chapter is twofold. Firstly, multiset transition systems are investigated as a model for concurrency used for representing the behaviour of Petri nets. Secondly, multiset transition systems serve as a general framework for defining notions to be used also in later chapters.

In Section 3.1 multiset transition systems are introduced. Then in Section 3.2 the notion of a (generalized) region is defined. In Section 3.3 a multiset transition system is associated with each Petri net, its *marking diagram*. Then a characterization is given of those multiset transition systems, called *PN-transition systems*, which are isomorphic to such a marking diagram. This characterization is based on the notion of a region. In Section 3.4 a characterization is given of those multiset transition systems which are isomorphic to the marking diagram of a 1-safe Petri net. It is shown that an equivalent representation of these multiset transition systems can be obtained using *asynchronous transition systems*. Finally, in Section 3.5 the characterization of PN-transition systems from Section 3.3 is used to express the relationship between multiset transition systems and Petri nets in a categorical framework.

This chapter mainly summarizes results from Mukund [63, 64] where missing proofs can be found.

3.1 Multiset Transition Systems

In this section we consider transition systems in which the transitions are labelled with multisets of actions over a given alphabet. In this way concurrency between actions is represented explicitly in the transition relation. The occurrence of the empty multiset is meant to denote inactivity at a state.

Definition 3.1.1

(1) A *multiset transition diagram* is a triple $(Q, X, \longrightarrow)$ where Q is a set of *states*, X is an alphabet of *actions*, and $\longrightarrow \subseteq Q \times M_F(X) \times Q$ is a *multiset transition relation* such that

$$(q, \underline{0}, q') \in \longrightarrow \Leftrightarrow \ q = q'.$$

(2) A *multiset transition system* is a quadruple $(Q, X, \longrightarrow, q_{in})$ where $(Q, X, \longrightarrow)$ is a multiset transition diagram and $q_{in} \in Q$ is an *initial state*. □

In contrast to [52, 63] we do not impose any conditions at this moment guaranteeing reachability of states and/or substep properties.

For a multiset transition diagram $TD = (Q, X, \longrightarrow)$ we write $q \xrightarrow{u} q'$ rather than $(q, u, q') \in \longrightarrow$. For $q, q' \in Q$ and $\rho \in (M_F(X))^+$ we write $q \xrightarrow{\rho} q'$, if there exist $q_0, \ldots, q_n \in Q$ and $u_1, \ldots, u_n \in M_F(X)$ where $n \geq 1$, such that

- $q_0 = q$ and $q_n = q'$ and
- $\rho = u_1 \ldots u_n$ and
- $\forall 1 \leq i \leq n. \, q_{i-1} \xrightarrow{u_i} q_i$.

We write $q \xrightarrow{\rho}$ if there exists $q' \in Q$ such that $q \xrightarrow{\rho} q'$.

An important subclass of multiset transition systems is obtained when all transitions are labelled with multisets containing at most one element.

Definition 3.1.2

(1) A *sequential transition diagram* is a multiset transition diagram $(Q, X, \longrightarrow)$ such that

$$q \xrightarrow{u} q' \Rightarrow |u| \leq 1.$$

(2) A *sequential transition system* is a multiset transition system $(Q, X, \longrightarrow, q_{in})$ such that $(Q, X, \longrightarrow)$ is a sequential transition diagram. □

Note that a multiset transition system could be viewed as a sequential transition system which has $M_F(X)$ as its underlying alphabet. One of the reasons why we do not take this point of view is that we want morphisms between multiset transition systems to preserve the actions in the multisets in a consistent way.

Morphisms between multiset transition systems should capture the fact that transitions in one multiset transition system can be "simulated" in the other.

Definition 3.1.3

Let $TS_i = (Q_i, X_i, \longrightarrow_i, q_i)$, $i = 1, 2$, be a pair of multiset transition systems. An *MTS-morphism* from TS_1 to TS_2 is a pair (f, g) with $f : X_1 \to X_2$ a partial function and $g : Q_1 \to Q_2$ a total function such that

(1) $g(q_1) = q_2$

(2) $q \xrightarrow{u}_1 q' \Rightarrow g(q) \xrightarrow{f(u)}_2 g(q')$. □

In [68] G-morphisms between sequential transition systems are introduced. When restricted to sequential transition systems, MTS-morphisms coincide with these G-morphisms.

For a multiset transition system TS let id_{TS} denote the *identity MTS-morphism* which is the pair of identity functions on the states and the actions of TS, respectively. Then an MTS-morphism (f, g) from TS_1 to TS_2 is an *MTS-isomorphism* iff there exists an MTS-morphism (f', g') from TS_2 to TS_1 such that $(f' \circ f, g' \circ g) = id_{TS_1}$ and $(f \circ f', g \circ g') = id_{TS_2}$.

3.2 Regions

Because they only have a notion of a global state, transition systems are more abstract than Petri nets. In this section (generalized) regions of multiset transition diagrams are defined in order to be able to decompose global states of multiset transition diagrams into local states. The results in this section will be used throughout the thesis.

In [22] regions have been introduced in the context of (partial) 2-structures in order to characterize the state spaces of elementary net systems. These regions are called *elementary regions* here and they are defined, as in [68], in the context of sequential transition diagrams.

Definition 3.2.1

Let $TD = (Q, X, \longrightarrow)$ be a sequential transition diagram. An *elementary region* of TD is a set $r \subseteq Q$ satisfying the following conditions.

(1) $(q \xrightarrow{a} q' \text{ and } q \in r \text{ and } q' \notin r) \Rightarrow \forall q_1 \xrightarrow{a} q_1'.\, (q_1 \in r \text{ and } q_1' \notin r)$.

(2) $(q \xrightarrow{a} q' \text{ and } q \notin r \text{ and } q' \in r) \Rightarrow \forall q_1 \xrightarrow{a} q_1'.\, (q_1 \notin r \text{ and } q_1' \in r)$.

An elementary region r of TD is *non-trivial* if $r \neq \emptyset$ and $r \neq Q$. □

The notion of an elementary region for sequential transition diagrams is lifted to sequential transition systems in the obvious way. Thus given a sequential transition system $TS = (Q, X, \longrightarrow, q_{in})$, we refer to the elementary regions of its underlying sequential transition diagram $(Q, X, \longrightarrow)$ as the elementary regions of TS. We use $e\mathcal{R}_{TS}$ to denote the set of non-trivial elementary regions of a sequential transition system TS.

An elementary region of a sequential transition system $TS = (Q, X, \longrightarrow, q_{in})$ is a set of states such that transitions labelled with the same action have the same crossing relation with respect to this set: either they are all leaving, or they are all entering, or they are not crossing the set at all. Let for $a \in X$,

$$^\circ a = \{r \in e\mathcal{R}_{TS} \mid \exists q \stackrel{a}{\longrightarrow} q'.(q \in r \text{ and } q' \notin r)\}$$

be the set of *input regions* of a and

$$a^\circ = \{r \in e\mathcal{R}_{TS} \mid \exists q \stackrel{a}{\longrightarrow} q'.(q \notin r \text{ and } q' \in r)\}$$

be the set of *output regions* of a.

Example 3.2.2

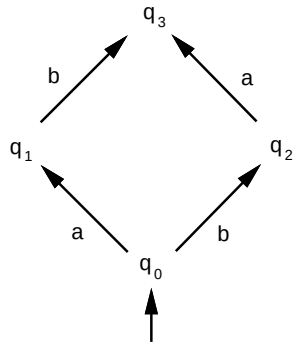


Figure 3.1: A sequential transition system TS

Let TS be the sequential transition system depicted in Figure 3.1. The elementary regions of TS are the trivial elementary regions $\emptyset$ and $\{q_0, q_1, q_2, q_3\}$, the input region $\{q_0, q_1\}$ of b, the input region $\{q_0, q_2\}$ of a, the output region $\{q_2, q_3\}$ of b, and the output region $\{q_1, q_3\}$ of a. □

Elementary regions of sequential transition systems are closely related to conditions of elementary net systems. In fact, with each sequential transition system TS an elementary net system can be associated which has the non-trivial elementary regions of TS as its conditions and has as its events those actions a for which $^\circ a \cup a^\circ \neq \emptyset$. The flow relation is such that for each event a, $^\bullet a = {}^\circ a$ and $a^\bullet = a^\circ$. Finally, the initial case consists of those non-trivial elementary regions which contain the initial state. It is interesting to note that the elementary net system obtained in this way

is contact-free. This follows easily from the fact that if r is a non-trivial elementary region of TS, then its complement $Q - r$ is also a non-trivial elementary region of TS.

In this construction of an elementary net system from a sequential transition system, the (global) states of the transition system are viewed as being composed of non-trivial regions. The regions containing a state are interpreted as conditions that hold in that state.

In the case of Petri nets, local states are more complex. Instead of being true or false, a local state is a place with an arbitrary number of tokens. Moreover, the relationship between a place and a transition of a Petri net is now a pair of natural numbers as given by the weight function.

These observations lead to the following definition of a *(generalized) region* [63, 40] as a multiset of states such that each action has a fixed effect on these multisets. In contrast to elementary regions, generalized regions are defined for arbitrary multiset transition diagrams.

Definition 3.2.3

Let $TD = (Q, X, \longrightarrow)$ be a multiset transition diagram. A *(generalized) region* of TD is a function $r : Q \cup X \to \mathbf{N} \cup (\mathbf{N} \times \mathbf{N})$ satisfying the following conditions.

(1) $\forall q \in Q.\, r(q) \in \mathbf{N}$ and $\forall a \in X.\, r(a) \in \mathbf{N} \times \mathbf{N}$.

For $a \in X$ we write $r(a) = ({}^r a, a^r)$.

(2) $q \xrightarrow{u} q' \Rightarrow (r(q) \geq \sum_{a \in X} u(a) \cdot {}^r a$ and $r(q') = r(q) + \sum_{a \in X} u(a) \cdot (a^r - {}^r a))$.

A generalized region r of TD is *non-trivial* if

$$\exists a \in X.\, r(a) \neq (0, 0).$$

$\mathcal{R}_{TD}$ is the set of non-trivial regions of TD. □

Similar to the situation for sequential transition diagrams, the notion of a generalized region is lifted to multiset transition systems. Thus given a multiset transition system $TS = (Q, X, \longrightarrow, q_{in})$, we refer to the (generalized) regions of its underlying multiset transition diagram $(Q, X, \longrightarrow)$ as the (generalized) regions of TS, and we use $\mathcal{R}_{TS}$ to denote the set of non-trivial regions of TS.

That the notion of a (generalized) region is indeed a generalization of an elementary region can be seen as follows. Let r be an elementary region of a sequential transition diagram $TD = (Q, X, \longrightarrow)$. Then r corresponds to the generalized region r' of TD which is defined as follows:

(1) $\forall q \in Q.\, r'(q) = \begin{cases} 1 & \text{if } q \in r \\ 0 & \text{if } q \notin r \end{cases}$

(2) $\forall a \in X.\, r'(a) = \begin{cases} (1,0) & \text{if } r \in {}^\circ a \\ (0,1) & \text{if } r \in a^\circ \\ (0,0) & \text{otherwise.} \end{cases}$

Example 3.2.4

Consider again the sequential transition system TS depicted in Figure 3.1. This transition system has an infinite number of regions. Some examples of regions are given in the following table:

region	a	b	q_0	q_1	q_2	q_3
r_1	$(1,0)$	$(0,0)$	1	0	1	0
r_2	$(0,0)$	$(1,0)$	2	2	1	1
r_3	$(1,2)$	$(0,0)$	3	4	3	4
r_4	$(1,2)$	$(2,6)$	3	4	7	8
r_5	$(1,1)$	$(1,1)$	1	1	1	1

The region r_1 in this table corresponds to the elementary region $\{q_0, q_2\}$. □

There is now a natural way to associate a Petri net with a multiset transition system by taking for the places of the Petri net the non-trivial regions of the multiset transition system.

Definition 3.2.5

Let $TS = (Q, X, \longrightarrow, q_{in})$ be a multiset transition system. The *Petri net associated with* TS is $tn(TS) = (\mathcal{R}_{TS}, X, W, M_{in})$ where

- $W : (\mathcal{R}_{TS} \times X) \cup (X \times \mathcal{R}_{TS}) \to \mathbf{N}$ is such that

$$\forall r \in \mathcal{R}_{TS}. \forall a \in X. (W(r,a) = {}^r a \text{ and } W(a,r) = a^r)$$

- $M_{in} : \mathcal{R}_{TS} \to \mathbf{N}$ is such that

$$\forall r \in \mathcal{R}_{TS}. M_{in}(r) = r(q_{in}).$$

□

Actually, we have assumed in the definition of $tn(TS)$ that $\mathcal{R}_{TS} \cap X = \emptyset$. We can do this without loss of generality, because otherwise we could work with a Petri net obtained by indexing every place $r \in \mathcal{R}_{TS}$ in an appropriate way. For this new Petri net similar results can be derived. We will ignore this possible notational complication here.

Example 3.2.6

In Figure 3.2 the multiset transition system TS from Figure 3.1 is depicted together with a part of the (infinite) Petri net $tn(TS)$. The places of $tn(TS)$ which are drawn are the regions of TS from Example 3.2.4 and the non-trivial elementary regions of TS (see also Example 3.2.2). The elementary regions are indicated as dotted circles. □

For a state $q \in Q$ of a multiset transition system $TS = (Q, X, \longrightarrow, q_{in})$, define the marking $M_q \in \mathcal{M}_{tn(TS)}$ by:

$$\forall r \in \mathcal{R}_{TS}. M_q(r) = r(q).$$

Then the dynamic behaviour of $tn(TS)$ is related to that of TS in the following way.

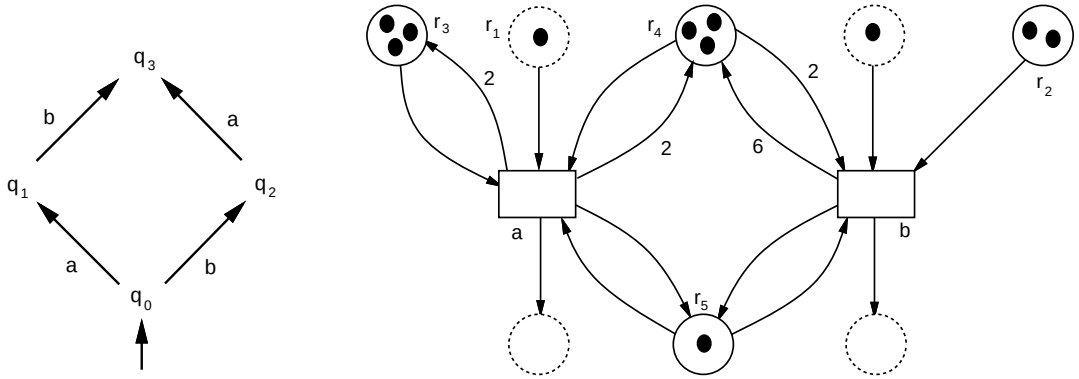


Figure 3.2: A sequential transition system with its associated Petri net

Lemma 3.2.7

Let $TS = (Q, X, \longrightarrow, q_{in})$ be a multiset transition system with associated Petri net $tn(TS) = (\mathcal{R}_{TS}, X, W, M_{in})$. Then

(1) $q \xrightarrow{u} q' \Rightarrow M_q[u\rangle_{tn(TS)} M_{q'}$

(2) $q_{in} \xrightarrow{\rho} \Rightarrow \rho \in MFS_{tn(TS)}$.

Proof.

(1) Suppose $q \xrightarrow{u} q'$. Then by the definition of a region, for all $r \in \mathcal{R}_{TS}$, $M_q(r) = r(q) \geq \sum_{a \in X} u(a) \cdot {}^r a = \sum_{a \in X} u(a) \cdot W(r, a)$ and hence $M_q[u\rangle_{tn(TS)}$. Moreover, for all $r \in \mathcal{R}_{TS}$, $M_{q'}(r) = r(q') = r(q) + \sum_{a \in X} u(a) \cdot (a^r - {}^r a) = M_q(r) + \sum_{a \in X} u(a) \cdot (W(a, r) - W(r, a))$. Now we can conclude that $M_q[u\rangle_{tn(TS)} M_{q'}$.

(2) Follows immediately from (1) and the observation that $M_{q_{in}} = M_{in}$. □

As the following example shows, neither the converse of (1) nor the converse of (2) in the above lemma holds in general.

Example 3.2.8

Let $TS = (Q, X, \longrightarrow, q_{in})$ be the sequential transition system depicted in Figure 3.3 where $q_{in} = q_0$. In TS there is no upperbound on the number of $a's$ that can occur. It is easy to see that $r : Q \cup X \to \mathbf{N} \times \mathbf{N}$ is a region of TS iff $r(q_0) \geq {}^r a$, $r(q_0) \geq {}^r b$, $r(q) = r(q_0) + b^r - {}^r b$, and $\forall i \geq 0 . r(q_i) = r(q_0) + i \cdot (a^r - {}^r a)$. Consequently, for every region r of TS and $a \in X$, $a^r \geq {}^r a$.

The Petri net $tn(TS)$ is depicted in Figure 3.4 where only some of the infinite number of places are drawn.

From the characterization of the regions of TS given above it easily follows that for all $i \geq 0$, $r(q_i) \geq r(q_0) \geq {}^r b$ and hence $M_{q_i}[b\rangle_{tn(TS)}$. On the other hand, $q_i \xrightarrow{b}$ only holds for $i = 0$, so the converses of part (1) and (2) of Lemma 3.2.7 do not hold. □

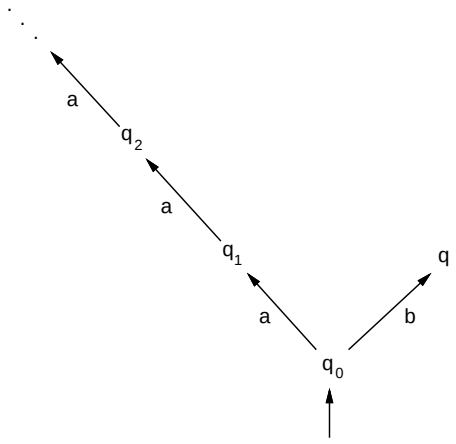


Figure 3.3: A sequential transition system TS

It is easy to see that for every multiset transition system TS, the Petri net $tn(TS)$ is S-simple and has no isolated places. Moreover, it is saturated in the sense that no new non-isolated places can be added without changing the behaviour (in terms of multiset firing sequences).

Next we show how to extend the map tn to MTS-morphisms in such a way that whenever ϕ is an MTS-morphism from TS_1 to TS_2, its image $tn(\phi)$ is a PN-morphism from $tn(TS_1)$ to $tn(TS_2)$. In the definition of this extension we use inverse regions.

So let $TS_i = (Q_i, X_i, \longrightarrow_i, q_i)$, $i = 1, 2$, be a pair of multiset transition systems and let $\phi = (f, g)$ be an MTS-morphism from TS_1 to TS_2. Define for a region r of TS_2, $\phi^{-1}(r) : Q_1 \cup X_1 \to \mathbf{N} \cup (\mathbf{N} \times \mathbf{N})$ by:

(1) $\forall q \in Q_1.\, \phi^{-1}(r)(q) = r(g(q))$

(2) $\forall a \in X_1.\, \phi^{-1}(r)(a) = \begin{cases} r(f(a)) & \text{if } f(a) \text{ is defined} \\ (0,0) & \text{otherwise.} \end{cases}$

Lemma 3.2.9

Let $TS_i = (Q_i, X_i, \longrightarrow_i, q_i)$, $i = 1, 2$, be multiset transition systems, let $\phi = (f, g)$ be an MTS-morphism from TS_1 to TS_2, and let r be a region of TS_2. Then $\phi^{-1}(r)$ is a region of TS_1. □

Proof.

Suppose $q \xrightarrow{u}_1 q'$. Then $g(q) \xrightarrow{f(u)}_2 g(q')$ by the definition of MTS-morphism. Since r is a region of TS_2 this implies that $\phi^{-1}(r)(q) = r(g(q)) \geq \sum_{b \in X_2} (f(u))(b) \cdot {}^{r}b = \sum_{a \in X_1} u(a) \cdot {}^{\phi^{-1}(r)}a$ and $\phi^{-1}(r)(q') = r(g(q')) = r(g(q)) + \sum_{b \in X_2} (f(u))(b) \cdot (b^{r} - {}^{r}b) = \phi^{-1}(r)(q) + \sum_{a \in X_1} u(a) \cdot (a^{\phi^{-1}(r)} - {}^{\phi^{-1}(r)}a)$. □

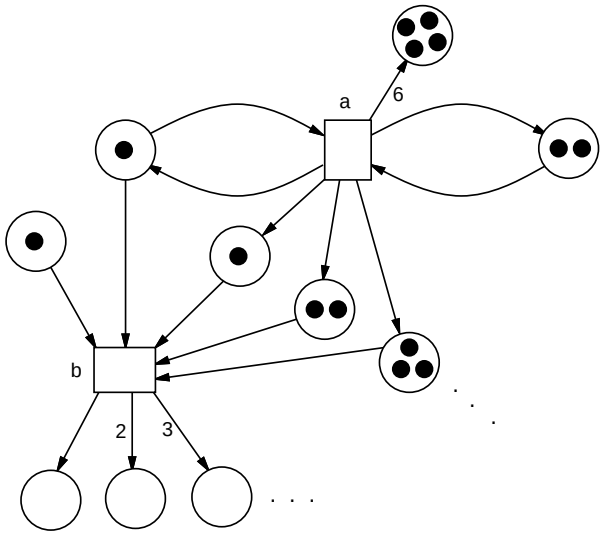


Figure 3.4: The Petri net $tn(TS)$

For elementary regions (viewed as generalized regions) this notion of inverse region coincides with the notion of the inverse of an elementary region as defined in [68].

Let TS_i, $i = 1, 2$, be a pair of multiset transition systems and let $\phi = (f, g)$ be an MTS-morphism from TS_1 to TS_2. Then define $tn(\phi) = (\beta_\phi, \eta_\phi)$ where $\eta_\phi = f$ and $\beta_\phi : \mathcal{R}_{TS_2} \to \mathcal{R}_{TS_1}$ is given by:

$$\beta_\phi(r) = \begin{cases} \phi^{-1}(r) & \text{if } \phi^{-1}(r) \text{ is non-trivial} \\ \text{undefined} & \text{otherwise.} \end{cases}$$

Lemma 3.2.10

Let $TS_i = (Q_i, X_i, \longrightarrow_i, q_i)$, $i = 1, 2$, be a pair of multiset transition systems and let $\phi = (f, g)$ be an MTS-morphism from TS_1 to TS_2. Then $tn(\phi) = (\beta_\phi, \eta_\phi)$ is a PN-morphism from $tn(TS_1) = (\mathcal{R}_{TS_1}, X_1, W_1, M_1)$ to $tn(TS_2) = (\mathcal{R}_{TS_2}, X_2, W_2, M_2)$.

Proof.

Let $r \in \mathcal{R}_{TS_2}$ be such that $\beta_\phi(r)$ is defined. Then $M_2(r) = r(q_2) = r(g(q_1)) = \phi^{-1}(r)(q_1) = M_1(\phi^{-1}(r))$. This proves condition (1) in the definition of a PN-morphism.

Let $a \in X_1$ be such that $\eta_\phi(a)$ is undefined. Then $f(a)$ is undefined, and therefore $\phi^{-1}(r)(a) = (0, 0)$ for all $r \in \mathcal{R}_{TS_2}$. Now assume that $r \in \mathcal{R}_{TS_2}$ is such that $r \in \beta_\phi^{-1}({}^\bullet a) \cup \beta_\phi^{-1}(a^\bullet)$. Thus $W_1(\beta_\phi(r), a) + W_1(a, \beta_\phi(r)) > 0$. Since $\beta_\phi(r) = \phi^{-1}(r)$, this leads to $\phi^{-1}(r)(a) = \beta_\phi(r)(a) = (W_1(\beta_\phi(r), a), W_1(a, \beta_\phi(r))) \neq (0, 0)$, a contradiction. Hence $\beta_\phi^{-1}({}^\bullet a) = \beta_\phi^{-1}(a^\bullet) = \emptyset$, which proves that $tn(\phi)$ satisfies condition (2) in the definition of a PN-morphism.

Finally, assume that $a \in X_1$ is such that $\eta_\phi(a) = f(a)$ is defined with $\eta_\phi(a) = b$. Then $(\beta_\phi(r))(a) = \phi^{-1}(r)(a) = r(f(a)) = ({}^r b, b^r)$ for all $r \in \mathcal{R}_{TS_2}$. Hence $r \in {}^\bullet b$ if and only if $\beta_\phi(r) \in {}^\bullet a$, that is $r \in \beta_\phi^{-1}({}^\bullet a)$. Similarly it can be proved that $\beta_\phi^{-1}(a^\bullet) = b^\bullet$.

Moreover, for all $r \in {}^\bullet b$, $W_1(\beta_\phi(r), a) = W_2(r, b)$ and, for all $r \in b^\bullet$, $W_1(a, \beta_\phi(r)) = W_2(b, r)$. This proves condition (3) in the definition of a PN-morphism. □

3.3 PN-Transition Systems

In the previous section we have seen how to associate a Petri net $tn(TS)$ with a multiset transition system TS. In this section we define for each Petri net a so-called *marking diagram*. Such a marking diagram is the multiset transition system induced by the firing rule. Marking diagrams are multiset transition systems, but not all multiset transition systems can be obtained as marking diagrams. A characterization is provided of those multiset transition systems that are marking diagrams (modulo isomorphism). The characterization of marking diagrams is in terms of regions.

This set-up from [63, 64] is again a generalization of the approach to the characterization of the state spaces of elementary net systems [22, 68].

Definition 3.3.1

Let $N = (S, T, W, M_{in})$ be a Petri net. The *marking diagram* of N is the multiset transition system $nt(N) = (\mathcal{RM}_N, T, \longrightarrow_N, M_{in})$ where

$$M \xrightarrow{u}_N M' \Leftrightarrow (M[u\rangle_N M' \text{ and } M, M' \in \mathcal{RM}_N).$$

□

In Figure 3.5 the Petri net N_1 from Figure 2.1 is depicted together with its marking diagram.

Definition 3.3.2

A multiset transition system TS is a *PN-transition system* if there exists a Petri net N such that $TS \equiv nt(N)$. □

The following three axioms, defined for an arbitrary multiset transition system $TS = (Q, X, \longrightarrow, q_{in})$, turn out to characterize the PN-transition systems.

(PT1) $\forall q \in Q.\, \exists \rho \in (M_F(X))^+.\, q_{in} \xrightarrow{\rho} q$.

(PT2) $(\forall r \in \mathcal{R}_{TS}.\, r(q) = r(q')) \Rightarrow q = q'$.

(PT3) $(\forall r \in \mathcal{R}_{TS}.\, r(q) \geq \sum_{a \in X} u(a) \cdot {}^r a) \Rightarrow q \xrightarrow{u}$.

The first axiom requires that every state is reachable, as is the case in the marking diagram of a Petri net where the states are the reachable markings.

The axiom (PT2) mirrors the property of the marking diagram of a Petri net that a state (i.e. reachable marking) is uniquely characterized by the number of tokens in each place.

By the definition of a region, the occurrence of a multiset of transitions implies that each region has enough "tokens" for this multiset to occur. The axiom (PT3) now states that the fact that each region contains enough tokens is not only necessary for a multiset of transitions to occur, but also sufficient. Thus this mirrors the property that the marking diagram of a Petri net contains *all* enablings at reachable markings.

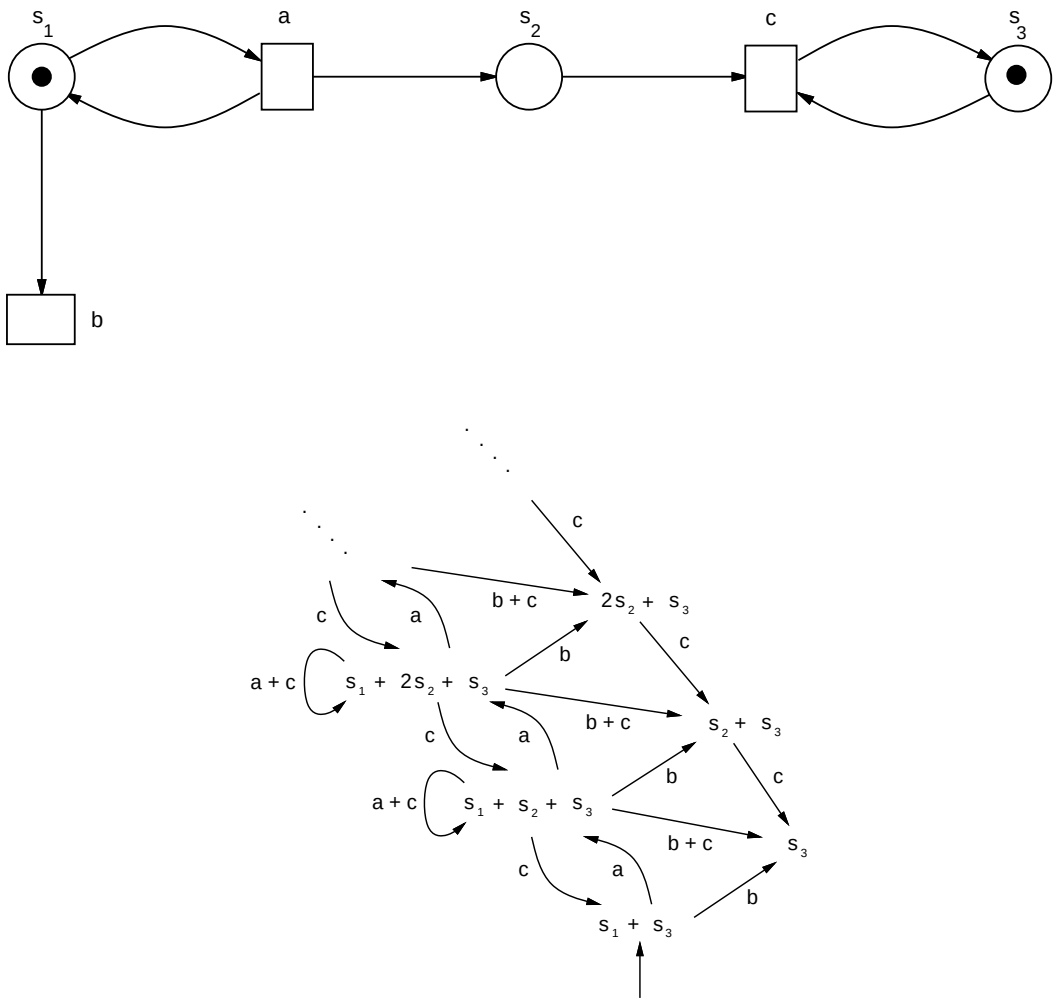


Figure 3.5: The Petri net N_1 with its marking diagram

Example 3.3.3

Consider the multiset transition system TS depicted in Figure 3.3. From the characterization of its regions in Example 3.2.8 it follows that for every $i \geq 1$ and for every $r \in \mathcal{R}_{TS}$, $r(q_i) \geq r(q_0) \geq {}^r b$. Axiom (PT3) would then imply that also $q_i \xrightarrow{b}$ for all $i \geq 1$. Hence TS does not satisfy (PT3).

The multiset transition system $nt(N_1)$ depicted in Figure 3.5 on the other hand, satisfies all three axioms (PT1), (PT2), and (PT3). □

The axioms (PT2) and (PT3) are quite strong. For instance, they guarantee that multisets can be split into arbitrary submultisets.

Lemma 3.3.4

Let $TS = (Q, X, \longrightarrow, q_{in})$ be a multiset transition system satisfying (PT1), (PT2), and (PT3). Then

$$q \xrightarrow{u} q' \Rightarrow \forall v \leq u.\, \exists q'' \in Q.\, (q \xrightarrow{v} q'' \text{ and } q'' \xrightarrow{u-v} q').$$

□

It is fairly easy to prove that every PN-transition system satisfies (PT1), (PT2), and (PT3). Using the map tn from multiset transition systems to Petri nets as defined in Section 3.2 it can be proved (see [63]) that the converse is also true.

Lemma 3.3.5

Let TS be a multiset transition system satisfying (PT1), (PT2), and (PT3). Then $TS \equiv nt(tn(TS))$. □

Hence we have the following characterization of PN-transition systems.

Theorem 3.3.6

A multiset transition system is a PN-transition system iff it satisfies the axioms (PT1), (PT2), and (PT3). □

Thus for every Petri net N there exists a saturated Petri net, namely $tn(nt(N))$, which has an isomorphic marking diagram. In a sense made more precise in Section 3.5 the Petri net $tn(nt(N))$ is the "best" representative among all Petri nets with the same (up to isomorphism) marking diagram.

Example 3.3.7

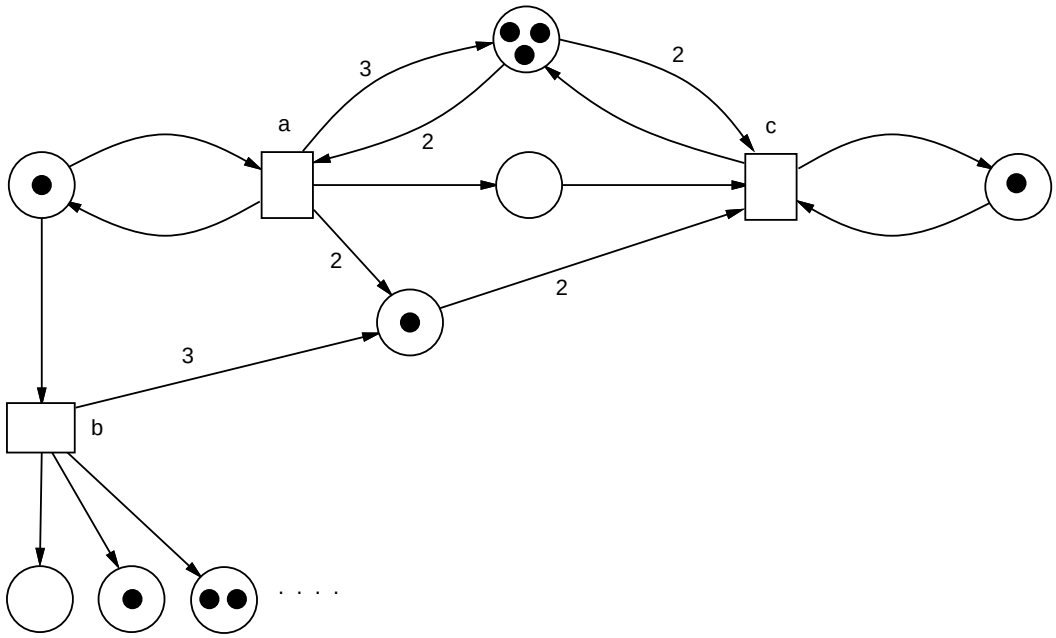


Figure 3.6: The Petri net $tn(nt(N_1))$

For the multiset transition system $nt(N_1)$ depicted in Figure 3.5, the Petri net $tn(nt(N_1))$ is drawn in Figure 3.6 where only some of the infinite number of places are given. Every region r of $nt(N_1)$ satisfies, e.g., the condition that $r(s_1 + s_3) = r(s_1 + s_2 + s_3) + c^r - {}^r c = r(s_1 + s_3) + a^r - {}^r a + c^r - {}^r c$ and hence ${}^r a + {}^r c = a^r + c^r$. Another condition which every region r of $nt(N_1)$ satisfies is that $a^r \geq {}^r a$. □

3.4 1-Safe PN-Transition Systems

In this section we investigate the relationship between 1-safe Petri nets and multiset transition systems. First a characterization is given of the PN-transition systems associated with 1-safe Petri nets. This characterization is from [64].

Definition 3.4.1
A PN-transition system TS is *1-safe* if there exists a 1-safe Petri net N such that $TS \equiv nt(N)$. □

To characterize the 1-safe PN-transition systems we need the notion of a *1-safe* region. Similar to how regions correspond to the places of an arbitrary Petri net, the *1-safe* regions correspond to the places of a 1-safe Petri net.

Definition 3.4.2
Let $TS = (Q, X, \longrightarrow, q_{in})$ be a multiset transition system. A region r of TS is *1-safe* if

$$\forall q \in Q.\, r(q) \in \{0,1\} \text{ and } \forall a \in X.\, r(a) \in \{(0,0),(0,1),(1,0),(1,1)\}.$$

$s\mathcal{R}_{TS}$ is the set of non-trivial 1-safe regions of TS. □

Note that the generalized regions corresponding to the elementary regions of a sequential transition system are 1-safe.

The following four axioms, formulated for an arbitrary multiset transition system $TS = (Q, X, \longrightarrow, q_{in})$, turn out to characterize the 1-safe PN-transition systems.

(PT1) $\forall q \in Q.\, \exists \rho \in (M_F(X))^+.\, q_{in} \xrightarrow{\rho} q$.

(PT2') $(\forall r \in s\mathcal{R}_{TS}.\, r(q) = r(q')) \Rightarrow q = q'$.

(PT3') $(\forall r \in s\mathcal{R}_{TS}.\, r(q) \geq \sum_{a \in X} u(a) \cdot {}^r a) \Rightarrow q \xrightarrow{u}$.

(PT4) $\forall q \in Q.\, \forall a \in X.\, \exists k \in \mathbf{N}.\, (q \xrightarrow{u} \Rightarrow u(a) < k)$.

The axioms (PT2) and (PT3) are strengthened by formulating them with respect to 1-safe regions. For 1-safe PN-transition systems the axiom (PT4) is required as a counterpart for condition (3) in the definition of 1-safe Petri nets, which prevents auto-concurrency. Now suppose TS satisfies (PT1), (PT2'), and (PT3'), but not (PT4). Then auto-concurrency in TS is still possible. If $q \xrightarrow{2 \cdot a}$, then, for every 1-safe region r of TS, $r(q) \geq 2 \cdot {}^r a$ and thus ${}^r a = 0$. Hence we also have that $r(q) \geq k \cdot {}^r a$ for all $k \in \mathbf{N}$. This implies by (PT3') that $q \xrightarrow{k \cdot a}$ for all $k \in \mathbf{N}$ and every 1-safe region r of TS. Hence in order to forbid auto-concurrency in TS, it is sufficient to forbid unbounded auto-concurrency at each state in TS as in (PT4).

It is easy to verify that the marking diagram of a 1-safe Petri net satisfies these four axioms. The proof of the converse is again based on a map from multiset transition systems to Petri nets. This map is defined by taking in the definition of the map tn only the 1-safe regions into account.

Definition 3.4.3

Let $TS = (Q, X, \longrightarrow, q_{in})$ be a multiset transition system which satisfies the axioms (PT1), (PT2'), (PT3'), and (PT4). Then $ts(TS) = (s\mathcal{R}_{TS}, X, W, M_{in})$ is the Petri net with

- $W : (s\mathcal{R}_{TS} \times X) \cup (X \times s\mathcal{R}_{TS}) \to \mathbf{N}$ is such that

$$\forall r \in s\mathcal{R}_{TS}. \forall a \in X. (W(r,a) = {}^{r}a \text{ and } W(a,r) = a^{r})$$

- $M_{in} : s\mathcal{R}_{TS} \to \mathbf{N}$ is such that

$$\forall r \in s\mathcal{R}_{TS}. M_{in}(r) = r(q_{in}).$$

□

Lemma 3.4.4

Let TS be a multiset transition system which satisfies the axioms (PT1), (PT2'), (PT3'), and (PT4). Then $ts(TS)$ is a 1-safe Petri net. □

For a multiset transition system TS which satisfies the three axioms (PT1), (PT2), and (PT3), Lemma 3.3.5 implies that the Petri net $tn(TS)$ yields a marking diagram which is isomorphic to TS. If TS satisfies the (stronger) axioms (PT1), (PT2'), (PT3'), and (PT4), then also the 1-safe Petri net $ts(TS)$ has a marking diagram isomorphic to TS.

Lemma 3.4.5

Let TS be a multiset transition system satisfying (PT1), (PT2'), (PT3'), and (PT4). Then $TS \equiv nt(ts(TS))$. □

This then leads to the following characterization of 1-safe PN-transition systems [64].

Theorem 3.4.6

A multiset transition system is a 1-safe PN-transition system iff it satisfies the axioms (PT1), (PT2'), (PT3'), and (PT4). □

To conclude this section we investigate an alternative representation of the marking diagram of a 1-safe Petri net.

In the marking diagram of an arbitrary Petri net every multiset of transitions which is enabled at a reachable marking is represented explicitly. By Lemma 2.1.10 however, concurrency in a 1-safe Petri net can also be derived from its sequential behaviour through a binary relation over its transitions. This leads to the model of asynchronous transition systems [86, 6] which can be used to give an equivalent representation of the marking diagram of a 1-safe Petri net.

Definition 3.4.7

An *asynchronous transition system* is a quintuple $TS = (Q, X, \Longrightarrow, q_{in}, Ind)$ where $(Q, X, \Longrightarrow, q_{in})$ is a sequential transition system and $Ind \subseteq X \times X$ is a symmetric, irreflexive *independence relation* which satisfy the following conditions.

(1) $a \in X \Rightarrow \exists q, q' \in Q.\, q \stackrel{a}{\Longrightarrow} q'$.

(2) $(q \stackrel{a}{\Longrightarrow} q'$ and $q \stackrel{a}{\Longrightarrow} q'') \Rightarrow q' = q''$.

(3) $((a,b) \in Ind$ and $q_1 \stackrel{a}{\Longrightarrow} q_2$ and $q_1 \stackrel{b}{\Longrightarrow} q_3) \Rightarrow \exists q_4 \in Q.\, (q_2 \stackrel{b}{\Longrightarrow} q_4$ and $q_3 \stackrel{a}{\Longrightarrow} q_4)$.

(4) $((a,b) \in Ind$ and $q_1 \stackrel{a}{\Longrightarrow} q_2$ and $q_2 \stackrel{b}{\Longrightarrow} q_4) \Rightarrow \exists q_3 \in Q.\, (q_1 \stackrel{b}{\Longrightarrow} q_3$ and $q_3 \stackrel{a}{\Longrightarrow} q_4)$. □

Condition (1) states that every action occurs in some transition and condition (2) demands that the underlying sequential transition system is deterministic. Conditions (3) and (4) capture the intuition behind the independence relation that two independent actions cannot prevent each others occurrence. Moreover, if two independent actions can occur at a given state after then the state reached after both of them have occurred, does not depend on the order of their occurrence.

In general an asynchronous transition system may contain independencies between actions which are never enabled at the same state. We say that an asynchronous transition system $TS = (Q, X, \Longrightarrow, q_{in}, Ind)$ is *reduced* iff

$$\forall (a,b) \in Ind.\, \exists q \in Q.\, (q \stackrel{a}{\Longrightarrow} \text{ and } q \stackrel{b}{\Longrightarrow}).$$

There is an obvious way to view each asynchronous transition system as a multiset transition system. Given an arbitrary asynchronous transition system $TS = (Q, X, \Longrightarrow, q_{in}, Ind)$, define $at(TS) = (Q, X, \longrightarrow, q_{in})$ where $\longrightarrow \subseteq Q \times P_F(X) \times Q$ is such that

$$(q \stackrel{u}{\longrightarrow} q' \text{ with } u = \{a_1, \ldots, a_n\} \text{ and } |u| = n) \Leftrightarrow$$

$$(q \stackrel{a_1 \ldots a_n}{\Longrightarrow} q' \text{ and } \forall 1 \leq i, j \leq n.\, (i \neq j \Rightarrow (a_i, a_j) \in Ind.))$$

Thus a step of actions can occur in $at(TS)$ at a given state iff the actions are pairwise independent and every possible interleaving of the actions can occur at this state.

In general, different asynchronous transition systems may be mapped to the same multiset transition system by at. When restricted to reduced asynchronous transition systems however, the map at is injective. A reduced asynchronous transition system TS can then be recovered uniquely from its multiset transition system representation under at: if $at(TS) = (Q, X, \longrightarrow, q_{in})$, then $TS = (Q, X, \Longrightarrow, q_{in}, Ind)$ where

- $q \stackrel{a}{\Longrightarrow} q' \Leftrightarrow q \stackrel{a}{\longrightarrow} q'$
- $Ind = \{(a,b) \mid a \neq b \text{ and } \exists q \in Q.\, q \stackrel{\{a,b\}}{\longrightarrow}\}$.

Next we show how to associate an asynchronous transition system with every 1-safe Petri net. Lemma 2.1.12 and Lemma 2.1.11 suggest the following definition of an independence relation to capture concurrency in a 1-safe Petri net.

Definition 3.4.8

Let $N = (S, T, W, M_{in})$ be a 1-safe Petri net. The *independence relation* $Ind_N \subseteq T \times T$ *associated with* N is given by:

$$Ind_N = \{(t_1, t_2) \in T \times T \mid t_1 \neq t_2 \text{ and } \exists M \in \mathcal{RM}_N . M[\{t_1, t_2\}\rangle_N\}.$$

□

Note that the independence relation associated with a 1-safe Petri net is irreflexive and symmetric.

For the 1-safe Petri nets N_3 and N_4 depicted in Figure 2.3 and Figure 2.4 we have that $Ind_{N_3} = \emptyset$ and $Ind_{N_4} = \{(a, b), (b, a)\}$.

Alternatively we could have given a *structural* definition of the independence relation associated with a 1-safe Petri net N as in, e.g., [98]:

$$Ind'_N = \{(t_1, t_2) \mid ({}^\bullet t_1 \cup t_1{}^\bullet) \cap ({}^\bullet t_2 \cup t_2{}^\bullet) = \emptyset\}.$$

The independence relation defined in this way also captures by Lemma 2.1.10 the concurrency in a 1-safe Petri net. On the other hand, two transitions of a 1-safe Petri net can have disjoint environments without ever being concurrent at some reachable marking. This is for instance the case for the transitions a and c of the Petri net depicted in Figure 3.7.

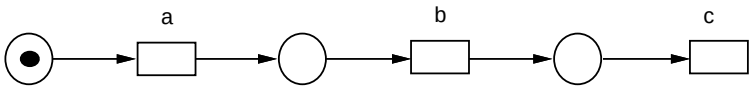


Figure 3.7: a and c have disjoint environments, but are never concurrent

For technical reasons we prefer to work with the minimal independence relation Ind_N which only contains independencies between transitions which can actually be enabled together.

Using this independence relation Ind_N there is an obvious way to associate a (reduced) asynchronous transition system with every 1-safe Petri net.

Definition 3.4.9

Let $N = (S, T, W, M_{in})$ be a 1-safe Petri net. Then the *asynchronous transition system associated with* N is $sa(N) = (\mathcal{RM}, T, \Longrightarrow_N, M_{in}, Ind_N)$, where

$$M \stackrel{u}{\Longrightarrow}_N M' \Leftrightarrow (|u| \leq 1 \text{ and } M \stackrel{u}{\longrightarrow}_N M').$$

□

From Lemma 2.1.10 it easily follows that this definition agrees (via the map at) with the definition of the marking diagram as given in Section 3.1.

Theorem 3.4.10
Let N be a 1-safe Petri net. Then $nt(N) \equiv at(sa(N))$. □

By the definition of Ind_N, the asychronous transition system $sa(N)$ is reduced. Hence at gives a bijection between the marking diagrams associated with 1-safe Petri nets and the asynchronous transition systems associated with 1-safe Petri nets. In other words, the asynchronous transition system associated with a 1-safe Petri net yields an equivalent representation of its marking diagram.

For elementary net systems it is even sufficient to associate a sequential transition system with it in order to capture its concurrent behaviour, because for this model arbitrary interleaving of occurrences implies that these occurrences are in fact concurrent.

However, for a 1-safe Petri net and also for a safe net system it is in general not possible to extract its concurrent transition system behaviour from its sequential transition system behaviour due to the behaviour in the presence of self-loops. Hence for 1-safe Petri nets concurrency must be represented explicitly in their associated transition system semantics.

Example 3.4.11
In Figure 3.8 the 1-safe Petri nets N_3 from Figure 2.3 and N_4 from Figure 2.4 are depicted together with their marking diagrams. Both marking diagrams have the same (up to isomorphism) sequential behaviour, but they differ in their concurrent behaviour. □

3.5 A Co-reflection Between $\mathcal{PTS}$ and $\mathcal{PN}$

In Section 3.3 the marking diagram of a Petri net is defined through the map nt. In this section it is shown that this map from Petri nets to PN-transition systems can be lifted to a functor which has a left adjoint.

The notion of morphism between multiset transition systems as defined in Section 3.1 leads to the following definition.

Definition 3.5.1
Let $\mathcal{PTS}$ be the category which has PN-transition systems as its objects and MTS-morphisms as its arrows. The identity morphism associated with an object TS is id_{TS}; the composition of MTS-morphisms (f, g) from TS_1 to TS_2 and (f', g') from TS_2 to TS_3 is the MTS-morphism $(f' \circ f, g' \circ g)$ from TS_1 to TS_3. □

By Lemma 2.2.2 PN-morphisms preserve occurrence sequences. Using this observation the map nt from Petri nets to PN-transition systems is now extended to a functor.

So let N_i, $i = 1, 2$, be a pair of Petri nets and let (β, η) be a PN-morphism from N_1 to N_2. Then define $nt((\beta, \eta)) = (f, g)$ where $f = \eta$ and $g : \mathcal{RM}_{N_1} \to \mathcal{RM}_{N_2}$ is given by $g(M) = \hat{M}$ where $\hat{M}$ is as defined in Lemma 2.2.2. Thus

$$g(M)(s) = \begin{cases} M(\beta(s)) & \text{if } \beta(s) \text{ is defined} \\ M_2(s) & \text{otherwise.} \end{cases}$$

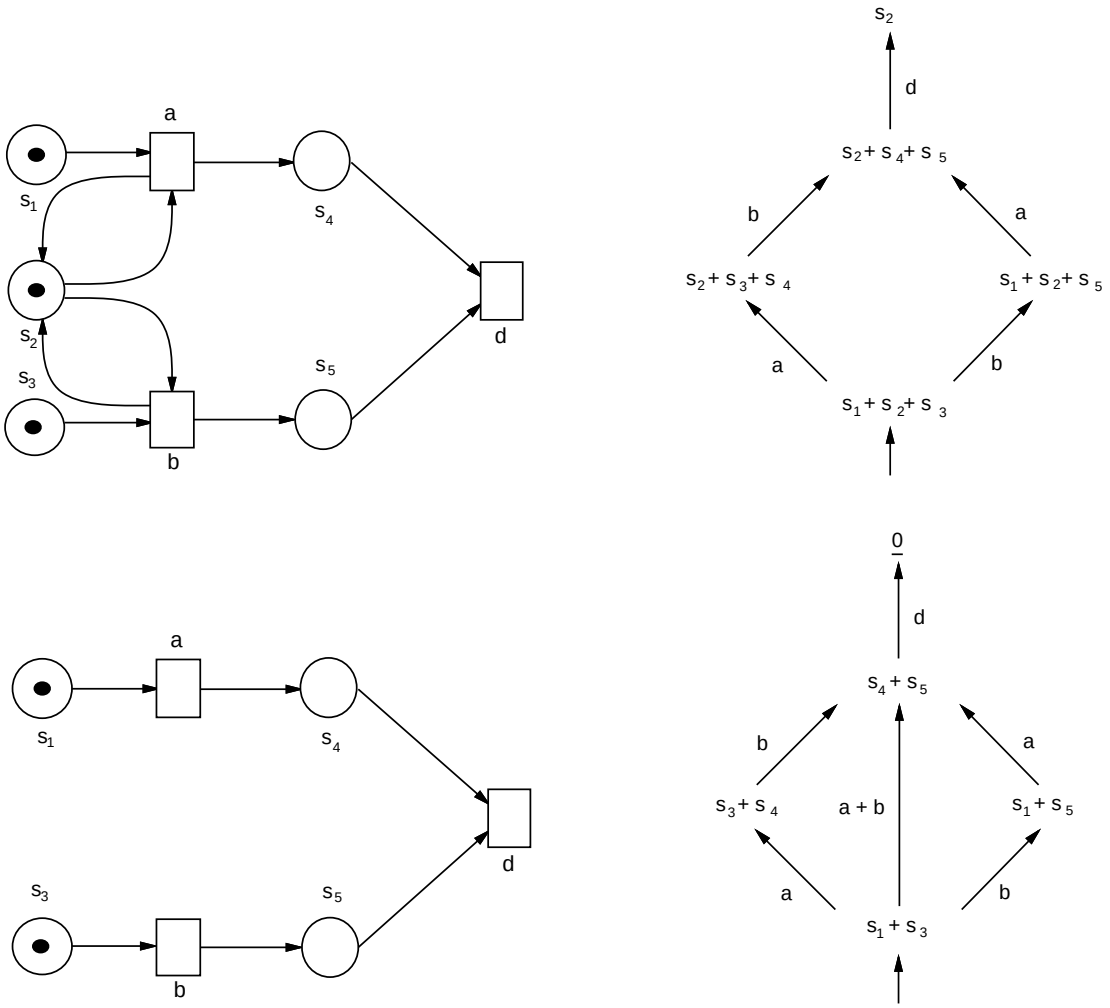


Figure 3.8: The Petri nets N_3 and N_4 with their marking diagrams

Note that by Lemma 2.2.2, $g(M)$ is indeed a reachable marking of N_2.

From Lemma 2.2.2 it also immediately follows that nt maps PN-morphisms to MTS-morphisms. Hence we have the following result.

Lemma 3.5.2

nt is a functor from $\mathcal{PN}$ to $\mathcal{PTS}$. □

In Section 3.2 a map tn has been defined, associating a Petri net with every multiset transition system. By Lemma 3.2.10, this construction can be extended to the corresponding morphisms.

The restriction of tn to the category $\mathcal{PTS}$ is also denoted by tn.

Lemma 3.5.3

tn is a functor from $\mathcal{PTS}$ to $\mathcal{PN}$. □

The fact that the category of PN-transition systems is more abstract than the category of Petri nets can now be phrased in categorical terms [63].

Theorem 3.5.4
$tn : \mathcal{PTS} \to \mathcal{PN}$ and $nt : \mathcal{PN} \to \mathcal{PTS}$ form a co-reflection with tn the left adjoint and the arrows id_{TS} as unit. □

The unit of this adjunction is indeed an MTS-isomorphism by Lemma 3.3.5. Thus starting with a PN-transition system TS, the marking diagram of the saturated Petri net $tn(TS)$ is MTS-isomorphic to TS. On the other hand, starting with a Petri net N, the (saturated) Petri net $tn(nt(N))$ associated with its marking diagram $nt(N)$ has itself a diagram which is isomorphic to $nt(N)$. Moreover, the co-unit of the adjunction gives for each Petri net N' which has the same (up to isomorphism) marking diagram as N a PN-morphism from $tn(nt(N))$ to N'.

Finally, a similar relationship can be established for the category of 1-safe PN-transition systems and the category of 1-safe Petri nets.

Definition 3.5.5
Let $\mathcal{PTS}s$ be the full subcategory of $\mathcal{PTS}$ the objects of which are 1-safe PN-transition systems. □

The restriction of nt to $\mathcal{PN}s$ is also denoted by nt.

In Section 3.4 the map ts from 1-safe PN-transition systems to 1-safe Petri nets is defined by associating with each such transition system a (1-safe) Petri net which has the 1-safe regions as its places. Similar to how the map tn from arbitrary multiset transition systems is extended to morphisms, the map ts is now extended to the corresponding morphisms.

Let TS_i, $i = 1, 2$, be a pair of 1-safe PN-transition systems and let $\phi = (f, g)$ be an MTS-morphism from TS_1 to TS_2. Then define $ts(\phi) = (\beta, \eta)$ where $\eta = f$ and $\beta : s\mathcal{R}_{TS_2} \to s\mathcal{R}_{TS_1}$ is given by:

$$\beta(r) = \begin{cases} \phi^{-1}(r) & \text{if } \phi^{-1}(r) \text{ is non-trivial} \\ \text{undefined} & \text{otherwise.} \end{cases}$$

Note that ts is well-defined because if r is a 1-safe region of TS_2, then $\phi^{-1}(r)$ is a region of TS_1 which is also 1-safe.

We then have the following result from [64] stating that the category of 1-safe PN-transition systems is a more abstract representation of the category of 1-safe Petri nets.

Theorem 3.5.6
$ts : \mathcal{PTS}s \to \mathcal{PN}s$ and $nt : \mathcal{PN}s \to \mathcal{PTS}s$ form a co-reflection with ts the left adjoint and the arrows id_{TS} as unit. □

A similar result has been proved independently in [98] where a co-reflection is established between the category of safe net systems and a subcategory of asynchronous transition systems.

Chapter 4

A Trace Semantics for Petri Nets

The dynamic behaviour of a Petri net is defined through the firing rule (Definition 2.1.2) in terms of transition occurrences and (reachable) markings. An (initial part of an) execution of the Petri net according to this rule and starting from the initial marking is here called a *run* of the Petri net. During a run there may be concurrency between (occurrences of) transitions, but conflicts between transitions are resolved.

Several notions for the representation of finite runs have already been discussed: occurrence sequences, multiset firing sequences, step firing sequences, and firing sequences. Each of these notions corresponds to a way of observing runs. They are non-branching in the sense that whenever a conflict occurs, a choice is made. Moreover, due to concurrency, one (finite) run of the Petri net may have several representations: a multiset of concurrent occurrences of transitions may be observed as distributed over varying submultisets in different orders. As a consequence each of these representations of the finite runs of a Petri net leads to a behavioural description of the Petri net which is unstructured in the sense that information on conflicts and concurrency is implicit, scattered, or not present at all.

The aim of this chapter is to provide a semantics for Petri nets based on finite runs, in which both the non-sequential and the branching aspects of their behaviour are captured. The approach we follow is inspired by the trace semantics for 1-safe Petri nets and elementary/safe net systems, initiated by Mazurkiewicz in [55], and further investigated in, e.g., [56, 80, 67, 98].

Using Mazurkiewicz' trace theory, the concurrency present in the behaviour of a 1-safe Petri net (or elementary/safe net system) is captured by the independence relation over the transitions from Definition 3.4.8. This independence relation induces an equivalence relation over the firing sequences. Each run of the system is represented by one of these equivalence classes, called *traces*. A prefix ordering of the traces yields a branching structure for the behaviour. This branching structure captures the relationship between the runs: two traces are ordered iff one of the corresponding runs is an initial part of the other.

In a general Petri net however, due to the multiplicity of tokens in places, concurrency and conflict are no longer global notions. Moreover, auto-concurrency has to be taken into account. To cater for these phenomena we generalize the notion of a trace. Having done this, it is easy to associate a poset of traces of the new kind with each

Petri net.

This chapter is organized as follows. In Section 4.1 we briefly recall the relevant notions from Mazurkiewicz' trace theory and the trace semantics for 1-safe Petri nets. In Section 4.2 we first discuss the generalizations which are needed to lift the trace semantics from the level of 1-safe Petri nets to the level of general Petri nets. This leads to the introduction of *local traces*. The trace semantics for Petri nets in terms of these local traces is given in Section 4.3. An axiomatic characterization of the resulting class of local trace languages is given in Section 4.4. Based on this characterization the relationship between local trace languages and Petri nets is expressed as a coreflection between the corresponding categories in Section 4.5. Then in Section 4.6 the relationship between local traces and Mazurkiewicz' traces is investigated in detail. A subclass of local trace languages is identified which corresponds to the class of Mazurkiewicz' trace languages, and it is shown that for 1-safe Petri nets the trace semantics in terms of Mazurkiewicz' traces agrees with the trace semantics in terms of local traces. Finally, in Section 4.7 we discuss some related work.

This chapter is based on [41], of which [40] is an extended abstract.

4.1 Mazurkiewicz' Traces and 1-Safe Petri Nets

This section introduces the basic notions from the theory of traces which has been initiated by Mazurkiewicz [55].

In order to avoid confusion after the local traces have been introduced in this chapter, all notions from the classical trace theory are prefixed with an M.

The behaviour of a sequential system can be described by sequential observations which are sequences over its alphabet of actions. For concurrent systems such as (1-safe) Petri nets however, such a description fails to capture the concurrency in the system. By Lemma 2.1.11 however, for 1-safe Petri nets it is possible to represent concurrency by a binary relation over its transitions. This motivates the consideration of alphabets together with a binary relation representing concurrency.

Definition 4.1.1

(1) Let X be an alphabet. An *M-independence relation (over X)* is a symmetric, irreflexive relation $Ind \subseteq X \times X$.

(2) An *M-concurrency alphabet* is a pair (X, Ind) where X is an alphabet and Ind is an M-independence relation over X. □

The intuition behind the independence relation is that sequential observations which only differ in the order of adjacent, independent symbols cannot be distinguished from each other. This leads to an equivalence relation over sequences as given in the next definition. Similar to how sequences are ordered by a prefix ordering, also these equivalence classes of sequences are ordered by a prefix ordering.

Definition 4.1.2
Let (X, Ind) be an M-concurrency alphabet.

(1) $\dot{-}_{Ind} \subseteq X^* \times X^*$ is given by:

$$\rho \dot{-}_{Ind} \rho' \Leftrightarrow \exists \rho_1, \rho_2 \in X^*. \exists (a,b) \in Ind. (\rho = \rho_1 ab \rho_2 \text{ and } \rho' = \rho_1 ba \rho_2).$$

(2) $\simeq_{Ind} \subseteq X^* \times X^*$, the *M-equivalence relation induced by* (X, Ind), is the least equivalence relation containing $\dot{-}_{Ind}$.

(3) For $\rho \in X^*$, $[\rho]_{Ind} = \{\rho' \in X^* \mid \rho \simeq_{Ind} \rho'\}$ is the *M-trace* (*over* (X, Ind)) containing ρ.

(4) $\preceq_{Ind} \subseteq (X^*/\simeq_{Ind}) \times (X^*/\simeq_{Ind})$ is the *M-trace ordering relation* (*over* (X, Ind)) given by:

$$[\rho]_{Ind} \preceq_{Ind} [\rho']_{Ind} \Leftrightarrow \exists \sigma \in X^*. \rho\sigma \simeq_{Ind} \rho'.$$

(5) An *M-trace language* (*over* (X, Ind)) is a subset of $X^*/\simeq_{Ind}$. □

If the M-concurrency alphabet (X, Ind) is clear from the context then we may omit the subscript $_{Ind}$ in the above defined notions.

An alternative characterization of the equivalence relation $\simeq_{Ind}$ is given in the following lemma from [1].

Lemma 4.1.3
Let (X, Ind) be an M-concurrency alphabet and let $\rho, \rho' \in X^*$. Then $\rho \simeq \rho'$ iff

(1) $\forall a \in X. num_a(\rho) = num_a(\rho')$ and

(2) $\forall (a,b) \in (X \times X) - Ind. proj_{\{a,b\}}(\rho) = proj_{\{a,b\}}(\rho')$. □

Thus two sequences ρ and ρ' are equivalent if and only if all symbols have the same number of occurrences in ρ and ρ' and the order of the occurrences of mutually dependent symbols is the same.

Given an M-concurrency alphabet (X, Ind) we can define, similar to the concatenation of ordinary sequences, for M-traces $[\rho]$ and $[\rho']$ over (X, Ind) their concatenation $[\rho] \circ [\rho'] = [\rho\rho']$. From Lemma 4.1.3 it easily follows that this concatenation operation is well-defined.

Observe that for each M-trace language TL over some M-concurrency alphabet (X, Ind) its *underlying language* $\{\rho \in X^* \mid [\rho] \in TL\}$ is *consistent*.

Definition 4.1.4
Let (X, Ind) be an M-concurrency alphabet and let $L \subseteq X^*$. Then L is *consistent* (*with respect to* (X, Ind)) if

$$\forall \rho \in L. [\rho] \subseteq L.$$

□

Conversely, every language $L \subseteq X^*$ which is consistent with respect to an M-concurrency alphabet (X, Ind), is the underlying language of the M-trace language $\{[\rho] \mid \rho \in L\}$ over (X, Ind).

From now on we specify an M-trace language over (X, Ind) with underlying language L as a triple (L, X, Ind). Note that in this way each M-trace language has a fixed alphabet X and a fixed M-independence relation Ind.

In general not every element from the M-independence relation of an M-trace language is needed in order to partition its underlying language into its M-traces. Those M-trace languages for which every element from the M-independence relation is needed are called *reduced* M-trace languages. These reduced M-trace languages will play a role in Section 4.6.

Definition 4.1.5

An M-trace language (L, X, Ind) is *reduced* if

$$\forall (a, b) \in Ind. \exists \rho ab\rho' \in L.$$

□

Morphisms between M-trace languages are defined as in [96, 98].

Definition 4.1.6

Let $TL_i = (L_i, X_i, Ind_i)$, $i = 1, 2$, be a pair of M-trace languages. An *MTL-morphism* from TL_1 to TL_2 is a partial function $f : X_1 \to X_2$ such that

(1) $\forall \rho \in L_1. f(\rho) \in L_2$

(2) $\forall (a, b) \in Ind_1. (f(a) \text{ and } f(b) \text{ are defined } \Rightarrow (f(a), f(b)) \in Ind_2).$ □

Thus MTL-morphisms preserve the underlying language and independence between actions.

For an M-trace language TL let id_{TL} denote the identity function on its alphabet. Then an MTL-morphism f from TL_1 to TL_2 is an *MTL-isomorphism* iff there exists an MTL-morphism g from TL_2 to TL_1 such that $g \circ f = id_{TL_1}$ and $f \circ g = id_{TL_2}$. Thus clearly, an MTL-morphism f from $TL_1 = (L_1, X_1, Ind_1)$ to $TL_2 = (L_2, X_2, Ind_2)$ is an MTL-isomorphism iff

(1) f is a bijection

(2) $\{f(\rho) \mid \rho \in L_1\} = L_2$

(3) $\forall a, b \in X_1. ((a, b) \in Ind_1 \Leftrightarrow (f(a), f(b)) \in Ind_2).$

It is easy to see that MTL-morphisms are behaviour-preserving in the following sense.

Lemma 4.1.7

Let f be an MTL-morphism from (L_1, X_1, Ind_1) to (L_2, X_2, Ind_2) and let $\rho, \rho' \in L_1$ be such that $\rho \dot{-}_{Ind_1} \rho'$. Then

$$f(\rho) \dot{-}_{Ind_2} f(\rho') \text{ or } f(\rho) = f(\rho').$$

□

Thus an MTL-morphism preserves the M-traces in an M-trace language.

In the above definition we use the convention given in Chapter 1 that for $\rho \in L_1$, $f(\rho)$ is the homorphic extension of the multiset extension of f, applied to ρ viewed as a multiset sequence. Thus on L_1, f is a total function.

As the last point in this section we define the M-trace semantics for 1-safe Petri nets [56, 98]. In doing this we use the independence relation Ind_N associated with a 1-safe Petri net $N = (S, T, W, M_{in})$ as defined in Definition 3.4.8. Recall that

$$Ind_N = \{(t_1, t_2) \in T \times T \mid t_1 \neq t_2 \text{ and } \exists M \in \mathcal{RM}_N.\, M[\{t_1, t_2\}\rangle_N\}.$$

The runs of a 1-safe Petri net are then represented by the M-traces generated by its firing sequences.

Definition 4.1.8

Let $N = (S, T, W, M_{in})$ be a 1-safe Petri net with associated independence relation Ind_N. Then

(1) $sm(N) = (FS_N, T, Ind_N)$ is the *M-trace language associated with* N

(2) $(sm(N), \preceq_{Ind_N})$ is the *M-trace behaviour* of N. □

It is easy to see that in the above definition FS_N is consistent with respect to (T, Ind_N) so that $sm(N)$ is well-defined. In part (2) of the above definition the restriction of $\preceq_{Ind_N}$ to $sm(N) \times sm(N)$ is also denoted by $\preceq_{Ind_N}$.

In what follows we write $\dot{-}_N$, $\simeq_N$, $[\rho]_N$, and $\preceq_N$, rather than $\dot{-}_{Ind_N}$, $\simeq_{Ind_N}$, $[\rho]_{Ind_N}$, and $\preceq_{Ind_N}$, respectively. If the 1-safe Petri net N is clear from the context then we may even omit the subscript N.

Note that by the definition of Ind_N, the M-trace language associated with a 1-safe Petri net N is reduced.

Example 4.1.9

In Figure 4.1 the 1-safe Petri nets N_3 and N_4 from Figure 2.3 and Figure 2.4 are depicted together with their M-trace behaviours. For both Petri nets abd and bad are firing sequences. Since $(a, b) \in Ind_{N_4}$, we have that $abd \simeq_{N_4} bad$. On the other hand, $Ind_{N_3} = \emptyset$, so that $abd \not\simeq_{N_3} bad$. □

Finally observe that, given a 1-safe Petri net N, we could have used in Definition 4.1.8 instead of Ind_N the structural definition of independence as in [56, 98] and mentioned before:

$$Ind'_N = \{(t_1, t_2) \mid ({}^\bullet t_1 \cup t_1{}^\bullet) \cap ({}^\bullet t_2 \cup t_2{}^\bullet) = \emptyset\}.$$

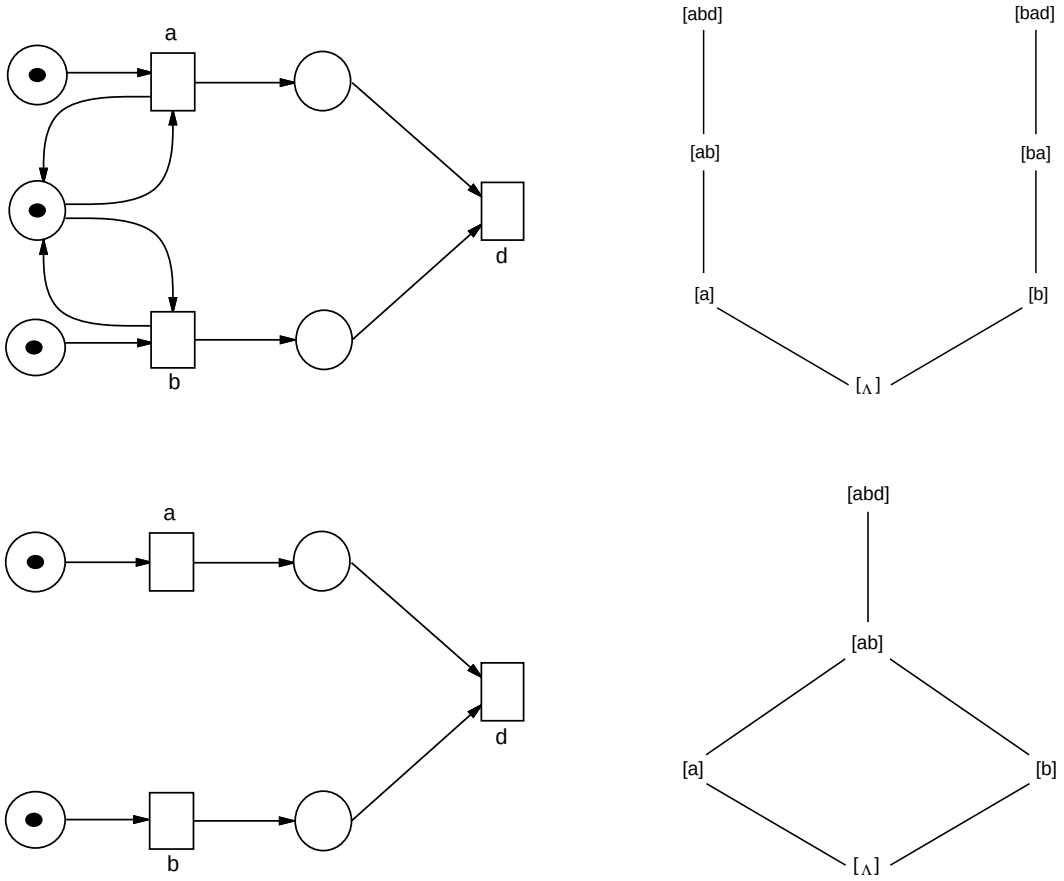


Figure 4.1: The Petri nets N_3 and N_4 with their M-trace behaviours

It is easy to see that this would not have affected the partitioning of FS into equivalence classes or their prefix-ordering, but the M-trace language would no longer be reduced. For technical reasons we prefer the independence relation to describe only those independencies between transitions which are concurrent at some reachable marking.

4.2 Local Traces

The M-trace semantics for 1-safe Petri nets explained in the previous section cannot be carried over directly to general Petri nets. Whereas concurrency in a 1-safe Petri net can be captured through a binary, global independence relation, for an arbitrary Petri net, concurrency is neither a global nor a binary property.

Now let us look at these problems for arbitrary Petri nets in some more detail.

The first problem is that concurrency is not a global notion for Petri nets: transitions which are concurrent at one reachable marking may be in conflict at another. As the Petri net N_6 depicted in Figure 4.2 shows, this makes it impossible to form traces from firing sequences through a global independence relation as in the previous section.

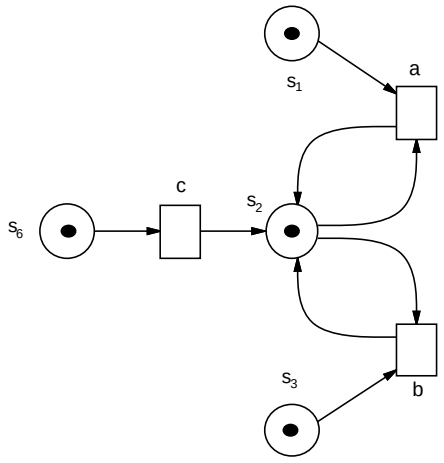


Figure 4.2: The Petri net N_6

For this Petri net both ab and ba are firing sequences. There are however not enough tokens for a and b to occur concurrently in the initial marking. Hence a and b should not be independent, and ab and ba should not be in the same trace. After the occurrence of c however, there are in the resulting marking enough tokens for the step $\{a, b\}$ to occur, so in this marking a and b should be independent, and cab and cba should be in the same trace.

Thus for general Petri nets we want the current marking to determine if transitions are independent. Equivalently, we can also say that the (multiset) firing sequence leading to this marking should determine the independence of transitions.

A second problem when attempting to lift the approach followed in Section 4.1 to the case of general Petri nets is that concurrency within arbitrary Petri nets cannot be characterized through a binary relation. For 1-safe Petri nets it follows from Lemma 2.1.11 that transitions in a set are concurrent at a marking iff each transition in the set is enabled individually and the transitions in the set are pairwise concurrent at this marking.

Now consider the Petri nets N_7 and N_8 depicted in Figure 4.3 and Figure 4.4. Both Petri nets have the same set of firing sequences and for both Petri nets each pair of transitions is concurrent at a reachable marking iff both transitions have not yet occurred. However, in N_8 the transitions a, b, and c can also occur all three concurrently at the initial marking, which is not possible for N_7.

Hence for general Petri nets the pairwise concurrency of transitions no longer implies the concurrency of the set as a whole. Moreover, it is now possible for a *multiset* of transitions to occur concurrently at a reachable marking due to auto-concurrency as for the Petri net N_2 depicted in Figure 2.2.

The problems pointed out above lead to the introduction in this section of a generalization of M-traces. The independencies in this more general set-up are defined *locally*, i.e. in a context-dependent fashion. In addition they specify, for chosen con-

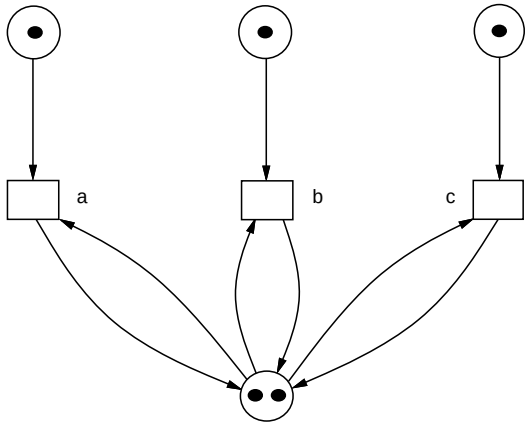


Figure 4.3: The Petri net N_7

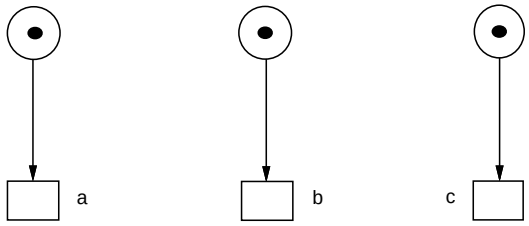


Figure 4.4: The Petri net N_8

texts, which (finite) multisets are independent. Moreover, the equivalence relation induced by these independencies is defined over multiset sequences.

Definition 4.2.1

(1) Let X be an alphabet. A *local independence relation* (*over* X) is a relation $I \subseteq (M_F(X))^+ \times M_F(X)$.

(2) A *local concurrency alphabet* is a pair (X, I), where X is an alphabet and I is a local independence relation over X. □

From now on we write L-concurrency alphabet and L-independence relation rather than local concurrency alphabet and local independence relation, respectively.

The basic notions from the classical trace theory are now lifted in the following way to the new setting.

Definition 4.2.2

Let (X, I) be an L-concurrency alphabet.

(1) $\doteq_I \subseteq (M_F(X))^+ \times (M_F(X))^+$ is given by:

$\rho \doteq_I \rho' \Leftrightarrow \exists \rho_1, \rho_2 \in (M_F(X))^+ . \exists u, v, u', v' \in M_F(X).$

[(1a) $\rho = \rho_1 uv\rho_2$ and $\rho' = \rho_1 u'v'\rho_2$ and

(1b) $u + v = u' + v'$ and

(1c) $(\rho_1, u + v) \in I$)].

(2) $\cong_I \subseteq (M_F(X))^+ \times (M_F(X))^+$, the *local trace equivalence relation induced by* (X, I), is the least equivalence relation containing $\doteq_I$.

(3) For $\rho \in (M_F(X))^+$, $[\rho]_I = \{\rho' \in (M_F(X))^+ \mid \rho \cong_I \rho'\}$ is the *local trace* containing ρ.

(4) $\lesssim_I \subseteq ((M_F(X))^+/\cong) \times ((M_F(X))^+/\cong)$ is the *local trace ordering relation* (*over* (X, I)) given by:

$$[\rho]_I \lesssim_I [\rho']_I \Leftrightarrow \exists \sigma \in (M_F(X))^+ . \rho\sigma \cong_I \rho'.$$

(5) A *local trace language* (*over* (X, I)) is a subset of $(M_F(X))^+/\cong_I$. □

From now on we also abbreviate the phrase local trace as L-trace. If the L-concurrency alphabet (X, I) is clear from the context we may omit the subscript $_I$ in the above defined notions.

Note that the ordering relation $\lesssim_I$ from Definition 4.2.2(4) is well-defined: whenever $\rho, \rho' \in (M_F(X))^+$ are such that $\rho \cong \rho'$, then $\rho\sigma \cong \rho'\sigma$ for all $\sigma \in (M_F(X))^+$.

Thus multiset sequences which are equivalent under the L-trace equivalence relation can be extended by an arbitrary multiset sequence leading to multiset sequences which are again equivalent. In the classical case such an operation can be lifted to a concatenation operation over traces given by $[\rho] \circ [\rho'] = [\rho\rho']$. However, due to the context-dependent nature of the L-independence relation, a similar operation is not well-defined for L-traces. Consider, for example, the concurrency alphabet $(\{a, b, c\}, \{(\underline{0}, \{b, c\})\})$; then $bc \cong cb$, but $abc \cong acb$ does not hold.

Definition 4.2.1 is a generous one in that no restrictions have been imposed on the L-independence relation. In applications, one might wish to place some suitable restrictions on the L-independence relation to capture the intended interpretation. For instance, where the L-trace languages are used to model the behaviour of distributed systems one might demand:

(D1) $(\rho, u) \in I \Rightarrow \forall v \leq u . (\rho, v) \in I$ and

(D2) $(\rho, u) \in I \Rightarrow \forall v \leq u . (\rho v, u - v) \in I$.

(D1) and (D2) would capture the intuition that $(\rho, u) \in I$ denotes the fact that the occurrences of actions mentioned in u are independent of each other at the state represented by ρ.

A third reasonable axiom one could demand would be:

(D3) $\rho \cong \rho' \Rightarrow ((\rho, u) \in I \Leftrightarrow (\rho', u) \in I)$.

This would ensure that I "agrees" with the equivalence relation $\cong$ induced by it.

However, we will not require such restrictions at this stage. The L-trace languages associated with Petri nets will all have L-independence relations satisfying (D1), (D2), and (D3).

Similar to the specification of M-trace languages we specify an L-trace language over (X, I) on the basis of its underlying language of multiset sequences in $(M_F(X))^+$.

Definition 4.2.3

Let (X, I) be an L-concurrency alphabet and let $L \subseteq (M_F(X))^+$. Then L is *consistent* (*with respect to* (X, I)) if

$$\forall \rho \in L.\, [\rho] \subseteq L.$$

□

An L-trace language TL over (X, I) is represented uniquely by the triple (L, X, I) where $L = \{\rho \mid [\rho] \in TL\}$ is its *underlying language*. A triple (L, X, I) with $L \subseteq (M_F(X))^+$ consistent with respect to (X, I) represents the L-trace language $\{[\rho] \mid \rho \in L\}$.

We now define morphisms between L-trace languages. Similar to the morphisms between M-trace languages, these morphisms are behaviour-preserving in the sense that they preserve the underlying language and independencies.

Definition 4.2.4

Let $TL_i = (L_i, X_i, I_i)$, $i = 1, 2$, be a pair of L-trace languages. An *LTL-morphism* from TL_1 to TL_2 is a partial function $f : X_1 \to X_2$ such that

(1) $\{f(\rho) \mid \rho \in L_1\} \subseteq L_2$

(2) $\{(f(\rho), f(u)) \mid (\rho, u) \in I_1\} \subseteq I_2$. □

For an L-trace language TL let id_{TL} denote the identity function on its alphabet. Then an LTL-morphism f from TL_1 to TL_2 is an *LTL-isomorphism* iff there exists an LTL-morphism g from TL_2 to TL_1 such that $g \circ f = id_{TL_1}$ and $f \circ g = id_{TL_2}$. Thus an LTL-morphism f from $TL_1 = (L_1, X_1, I_1)$ to $TL_2 = (L_2, X_2, I_2)$ is an LTL-isomorphism iff

(1) f is a bijection

(2) $\{f(\rho) \mid \rho \in L_1\} = L_2$

(3) $\{(f(\rho), f(u)) \mid (\rho, u) \in I_1\} = I_2$.

If TL_1 and TL_2 are LTL-isomorphic then we denote this by $TL_1 \equiv TL_2$.

Similar to the situation for MTL-morphisms, LTL-morphisms preserve equivalence of multiset sequences.

Lemma 4.2.5

Let f be an LTL-morphism from (L_1, X_1, I_1) to (L_2, X_2, I_2) and let $\rho, \rho' \in L_1$ be such that $\rho \doteq_{I_1} \rho'$. Then $f(\rho) \doteq_{I_2} f(\rho')$.

Proof.

Suppose $\rho = \rho_1 uv\rho_2$, $\rho' = \rho_1 u'v'\rho_2$, $u + v = u' + v'$, and $(\rho_1, u + v) \in I_1$. Then $(f(\rho_1), f(u) + f(v)) = (f(\rho_1), f(u + v)) \in I_2$ by the definition of LTL-morphism and hence $f(\rho) = f(\rho_1)f(u)f(v)f(\rho_2) \doteq_{I_2} f(\rho_1)f(u')f(v')f(\rho_2) = f(\rho')$. □

Finally note that concurrency must be preserved *locally* by LTL-morphisms. Consider, e.g., the L-trace language $TL = (L, X, I)$ with $X = \{a, b, c\}$, $L = (M_F(X))^+$ and $I = (M_F(X))^+ \times M_F(X)$, and the L-trace language $TL' = (L, X, I')$ with $I' = I - (\{a\}, \{b, c\})$. Although $\rho \cong_I \rho'$ iff $\rho \cong_{I'} \rho'$ for all $\rho, \rho' \in L$, the identity function on X is not an LTL-morphism from TL to TL'. Note that on the other hand this function is an LTL-morphism from TL' to TL.

4.3 L-Traces and Petri Nets

In this section a trace semantics for Petri nets is defined in terms of L-traces, similar to the definition of the M-trace semantics for 1-safe Petri nets. In order to be able to do this, we first need to associate with each Petri net an L-independence relation. This L-independence relation should capture the concurrency in the Petri net and be formulated in terms of multisets (of transitions). For this reason, instead of dealing with concurrency of a multiset of transitions at a reachable marking, we now deal with concurrent multisets after a given multiset firing sequence leading to a marking.

Definition 4.3.1

Let $N = (S, T, W, M_{in})$ be a Petri net.

(1) The *L-independence relation* $I_N \subseteq (M_F(T))^+ \times M_F(T)$ *associated with* N is the set

$$I_N = \{(\rho, u) \mid \rho u \in MFS\}.$$

(2) The *L-concurrency alphabet associated with* N is (T, I_N). □

In order to simplify the notation we write $\doteq_N$, $\cong_N$, $\lesssim_N$, and $[\rho]_N$ instead of $\doteq_{I_N}$, $\cong_{I_N}$, $\lesssim_{I_N}$, and $[\rho]_{I_N}$, respectively, in what follows.

It is easy to verify that the L-independence relation I_N associated with a Petri net N satisfies the properties

(D1) $(\rho, u) \in I_N \Rightarrow \forall v \leq u.\, (\rho, v) \in I_N$ and

(D2) $(\rho, u) \in I_N \Rightarrow \forall v \leq u.\, (\rho v, u - v) \in I_N$

mentioned in Section 4.2.

As the next lemma shows, all multiset sequences equivalent with a multiset firing sequence of a given Petri net are also multiset firing sequences of that Petri net.

Lemma 4.3.2

Let $N = (S, T, W, M_{in})$ be a Petri net. Then MFS is consistent with respect to (T, I_N).

Proof.

It is sufficient to prove that $\rho' \in MFS$ whenever $\rho' \in (M_F(T))^+$ is such that $\rho \doteq_N \rho'$ where $\rho = \rho_1 uv\rho_2 \in MFS$, $\rho' = \rho_1 u'v'\rho_2$, $u + v = u' + v'$, and $(\rho_1, u + v) \in I_N$. Since $\rho_1 uv\rho_2 \in MFS$, we also have $\rho_1 uv \in MFS$. Furthermore $(\rho_1 u', v') \in I_N$ because I_N satisfies the property (D2) mentioned above. Hence $\rho_1 u'v' \in MFS$ by the definition of I_N. Since $num_t(\rho_1 uv) = num_t(\rho_1 u'v')$ for all $t \in T$, $\rho_1 uv$ and $\rho_1 u'v'$ lead to the same marking, and so $\rho_1 uv\rho_2 \in MFS$ iff $\rho_1 u'v'\rho_2 \in MFS$. Since $\rho = \rho_1 uv\rho_2 \in MFS$ this implies $\rho' = \rho_1 u'v'\rho_2 \in MFS$. □

Lemma 4.3.2 implies that I_N also satisfies the property (D3) mentioned in Section 4.2:

(D3) $\rho \cong_N \rho' \Rightarrow ((\rho, u) \in I_N \Leftrightarrow (\rho', u) \in I_N)$.

The trace semantics for Petri nets is now defined as follows.

Definition 4.3.3

Let $N = (S, T, W, M_{in})$ be a Petri net. Then

(1) $nl(N) = (MFS_N, T, I_N)$ is the *L-trace language associated with* N.

(2) $(nl(N), \lesssim_N)$ is the *L-trace behaviour* of N. □

In part (2) of the above definition, the restriction of $\lesssim_N$ to $nl(N) \times nl(N)$ is also denoted by $\lesssim_N$.

By the following lemma every L-trace in the L-trace language associated with a Petri net contains at least one sequential representative, in which all multisets are singletons. Hence it is sufficient to consider the L-traces generated by the firing sequences.

Lemma 4.3.4

Let $N = (S, T, W, M_{in})$ be a Petri net and let $\rho \in MFS$. Then

$$\exists \rho' \in FS.\, \rho \cong \rho'.$$

Proof.

Suppose $\rho = \rho_1 u\rho_2$ with $|u| > 1$. Then by the definition of I_N, $(\rho_1, u) \in I_N$. Let $t \in T$ be such that $u(t) \geq 1$. Then $\rho \cong \rho_1(u - t)t\rho_2$. Repeatedly applying this reasoning yields the required $\rho' \in FS$ with $\rho \cong \rho'$. □

Example 4.3.5

The L-trace behaviour of the Petri net N_6 from Figure 4.2 is depicted in Figure 4.5. In N_6 independence of a and b is determined by the history (current marking). Since $(\underline{0}, \{a, b\}) \notin I_{N_6}$, the step firing sequences ab and ba are not equivalent (under $\cong_{N_6}$). On the other hand, $(c, \{a, b\}) \in I_{N_6}$. Thus $abc \cong_{N_6} bac$, because $abc \doteq_{N_6} acb \doteq_{N_6} cab \doteq_{N_6} cba \doteq_{N_6} bca \doteq_{N_6} bac$. In this reasoning we used apart from $(c, \{a, b\}) \in I_{N_6}$ also that $(a, \{b, c\}), (\underline{0}, \{a, c\}), (\underline{0}, \{b, c\}), (b, \{a, c\}) \in I_{N_6}$.

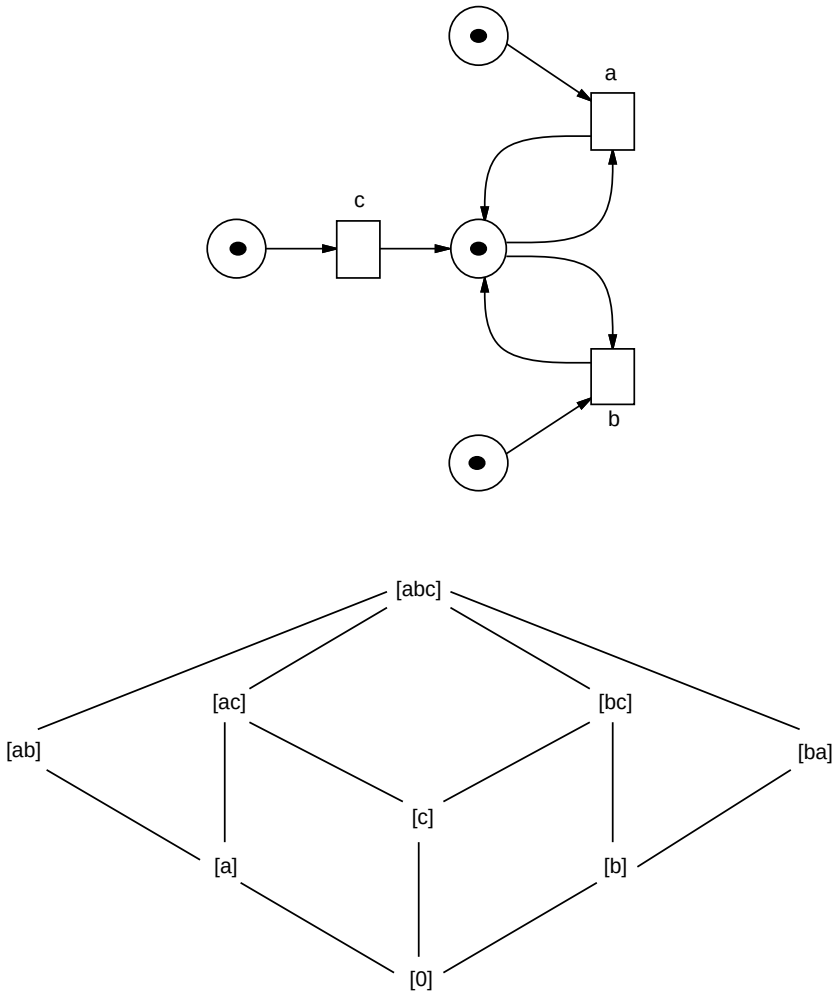


Figure 4.5: The Petri net N_6 with its L-trace behaviour

In Figure 4.6 a Petri net N_9 is depicted together with its L-trace behaviour. In the L-trace behaviour of N_9 the traces $[a]$ and $[c]$ have upperbounds $[ac]$ and $[abc]$, but they have no least upperbound. In other words, the L-trace behaviour of N_9 is not consistently complete. This contrasts with the situation for 1-safe Petri nets where the M-trace behaviour is always consistently complete.

Finally, the Petri net N_1 from Figure 2.1 and its L-trace behaviour are depicted in Figure 4.7. □

4.4 PN-Trace Languages

In this section we characterize those L-trace languages, called PN-trace languages, which are isomorphic to an L-trace language associated with a Petri net. The approach is similar to the approach followed in Section 3.3 where PN-transition systems are characterized.

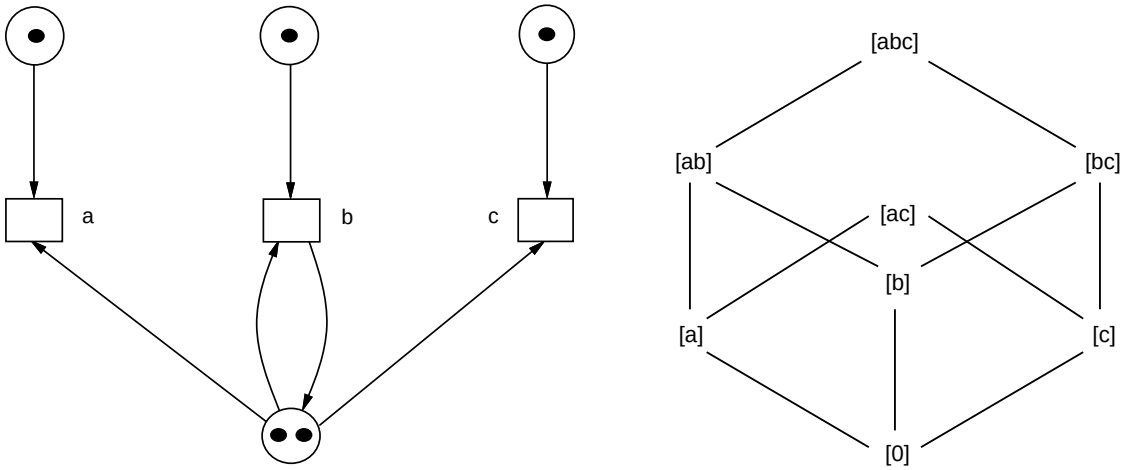


Figure 4.6: The Petri net N_9 with its L-trace behaviour

For L-trace languages four axioms are proposed of which it is easy to prove that every PN-trace language satisfies them. To prove the converse, with every L-trace language satisfying the four axioms a Petri net is associated by viewing the L-trace language as a multiset transition system and then applying the map *tn* defined in Section 3.2. Then the L-trace language generated by this Petri net and the original L-trace language are proved to be isomorphic.

Definition 4.4.1
An L-trace language TL is a *PN-trace language* if there exists a Petri net N such that $TL \equiv nl(N)$. □

One of the axioms appearing in the characterization of PN-trace languages is a regional axiom, which we use to capture the fact that *all* runs of a Petri net are represented in its associated L-trace language. In order to formulate this axiom we view L-trace languages as multiset transition diagrams. The regions of an L-trace language can then be defined in terms of this multiset transition diagram.

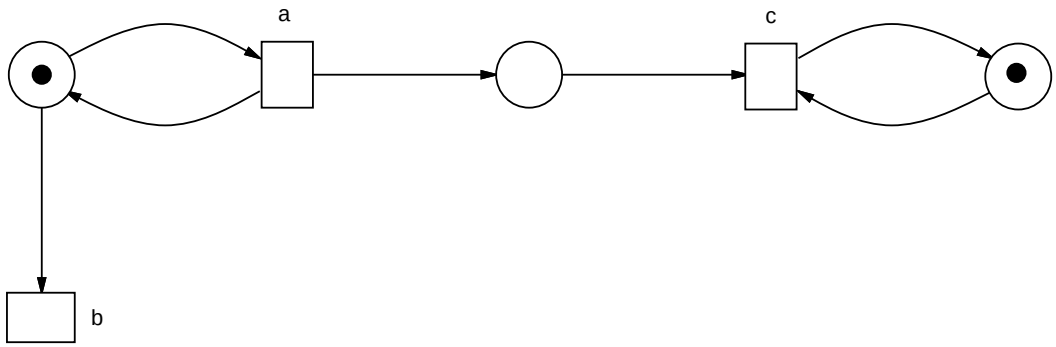
Let $TL = (L, X, I)$ be an L-trace language. Then TL gives rise to the multiset transition diagram $(L/\cong, X, \longrightarrow_{TL})$ where $\longrightarrow_{TL}$, the *multiset transition relation associated with* TL, is given by:

$$[\rho] \xrightarrow{u}_{TL} [\rho'] \Leftrightarrow \rho u \cong \rho'.$$

We refer to the regions of this multiset transition diagram as the regions of TL. Also we let $\mathcal{R}_{TL}$ denote the set of non-trivial regions of $(L/\cong, X, \longrightarrow_{TL})$.

So a region of TL is a function $r : L/\cong \cup X \to \mathbf{N} \cup (\mathbf{N} \times \mathbf{N})$ satisfying the following conditions.

(1) $\forall [\rho] \in L/\cong . r([\rho]) \in \mathbf{N}$ and $\forall a \in X. r(a) \in \mathbf{N} \times \mathbf{N}$.



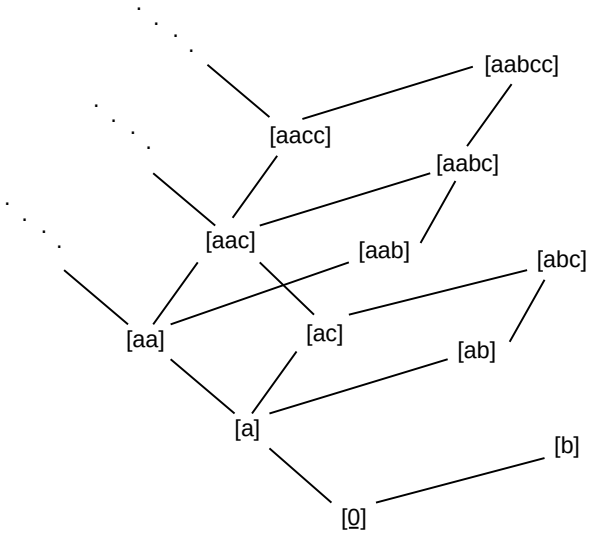


Figure 4.7: The Petri net N_1 with its L-trace behaviour

(2) $\rho u \cong \rho' \Rightarrow (r([\rho]) \geq \sum_{a \in X} u(a) \cdot {}^{r}a \text{ and } r([\rho']) = r([\rho]) + \sum_{a \in X} u(a) \cdot (a^{r} - {}^{r}a))$.

The four axioms that turn out to characterize the PN-trace languages can now be formulated. They are stated in terms of an arbitrary L-trace language $TL = (L, X, I)$.

(PL0) $L \neq \emptyset$.

(PL1) $\rho u \in L \Rightarrow \rho \in L$.

(PL2) $\rho u \in L \Leftrightarrow (\rho, u) \in I$.

(PL3) $\rho \in L \Rightarrow ((\forall r \in \mathcal{R}_{TL}.\, r([\rho]) \geq \sum_{a \in X} u(a) \cdot {}^{r}a) \Rightarrow \rho u \in L)$.

The first two axioms require that TL is non-empty and (its underlying language is) prefix-closed. Note that together they ensure that $\underline{0} \in L$. The third axiom states that L "agrees" with the L-independence relation I. Thus (PL2) together with (PL1) ensures

that the multisets in multiset sequences in L are all independent in the corresponding context. Moreover, (PL2) states that I is minimal in the sense that it contains no other independencies, and it also guarantees by the consistency of L a submultiset property in the sense that if $\rho u \in L$, then also $\rho v(u-v) \in L$ for every $v \leq u$. The final axiom (PL3) for characterizing the PN-trace languages corresponds to the axiom (PT3) for characterizing the PN-transition systems. This axiom captures the fact that all runs of a Petri net are represented in its associated L-trace language.

The characterization of PN-transition systems also mentions the following regional axiom:

(PT2) $(\forall r \in \mathcal{R}_{TS}.\, r(q) = r(q')) \Rightarrow q = q'$.

As the following example shows, we cannot have a similar axiom for characterizing the PN-trace languages.

Example 4.4.2

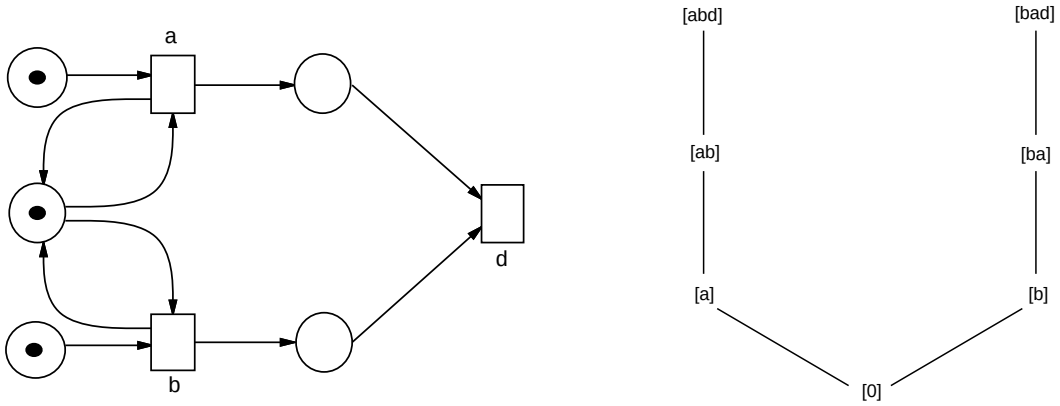


Figure 4.8: The Petri net N_3 with its L-trace behaviour

In Figure 4.8 the Petri net N_3 from Figure 2.3 is depicted together with its L-trace behaviour. For every $r \in \mathcal{R}_{nl(N_3)}$, $r([ab]) = r([\underline{0}]) + a^r + b^r - {}^r a - {}^r b = r([ba])$, but $ab \not\cong ba$. Hence the L-trace language associated with N_3 has two different L-traces at which every non-trivial region of $nl(N_3)$ has the same value. □

Now we turn to the proof that every PN-trace language satisfies these axioms. First of all, we prove in the following lemma that all axioms are preserved under LTL-isomorphisms.

Lemma 4.4.3

Let $TL_i = (L_i, X_i, I_i)$, $i = 1, 2$, be a pair of L-trace languages such that $TL_1 \equiv TL_2$. Then, for each $i = 0, 1, 2, 3$, TL_1 satisfies (PLi) iff TL_2 satisfies (PLi).

Proof.

Preservation of (PL0), (PL1), and (PL2) follows immediately from the definition of LTL-isomorphism.

In order to prove that TL_2 satisfies (PL3) if TL_1 does, let f be an LTL-isomorphism from TL_2 to TL_1. Suppose $\rho \in L_2$ and $\rho u \notin L_2$. Then we also have $f(\rho) \in L_1$ and $f(\rho u) \notin L_1$ by the definition of LTL-isomorphism. Hence by (PL3) there exists $r \in \mathcal{R}_{TL_1}$ such that $r([f(\rho)]_{I_1}) < \sum_{a \in X_1}(f(u))(a) \cdot {}^{r}a$. Define the function $r' : (L_2/\cong_{I_2}) \cup X_2 \to \mathbf{N} \cup (\mathbf{N} \times \mathbf{N})$ by:

(1) $\forall [\rho']_{I_2} \in L_2/\cong_{I_2} . r'([\rho']_{I_2}) = r([f(\rho')]_{I_1})$

(2) $\forall a \in X_2 . r'(a) = r(f(a))$.

Note that r' is well-defined by Lemma 4.2.5. Since by Lemma 4.2.5, $[f(\rho_1)]_{I_1} \xrightarrow{f(v)}_{TL_1} [f(\rho_2)]_{I_1}$ whenever $[\rho_1]_{I_2} \xrightarrow{v}_{TL_2} [\rho_2]_{I_2}$, it is easy to see that r' is a non-trivial region of TL_2. Moreover, $r'([\rho]_{I_2}) = r([f(\rho)]_{I_1}) < \sum_{a \in X_1}(f(u))(a) \cdot {}^{r}a = \sum_{b \in X_2}(f(u))(f(b)) \cdot {}^{r}f(b) = \sum_{b \in X_2} u(b) \cdot {}^{r'}b$. Hence TL_2 satisfies (PL3). Since $\equiv$ is symmetric, this completes our proof. □

In order to show that every PN-trace language satisfies (PL0) through (PL3), we now only have to prove that every L-trace language associated with a Petri net satisfies these four axioms.

As a first step in proving this, the following lemma shows that every place s of a Petri net $N = (S, T, W, M_{in})$ determines a region $r\langle s \rangle$ of its associated L-trace language $nl(N)$ in the following way. Define $r\langle s \rangle : (MFS/\cong) \cup T \to \mathbf{N} \cup (\mathbf{N} \times \mathbf{N})$ by:

(1) $\forall [\rho] \in MFS/\cong . r\langle s \rangle([\rho]) = M_\rho(s)$

(2) $\forall t \in T . r\langle s \rangle(t) = (W(s,t), W(t,s))$.

Thus for each $t \in T$, the function $r\langle s \rangle$ gives the number of tokens t takes from s and the number of tokens t puts in s and, for each $[\rho] \in MFS/\cong$, $r\langle s \rangle$ gives the number of tokens in s after the multiset firing sequence ρ. Note that $r\langle s \rangle$ is well-defined because if $\rho, \rho' \in MFS$ are such that $\rho \cong \rho'$, then $mset(\rho) = mset(\rho')$ and hence $M_\rho = M_{\rho'}$.

Lemma 4.4.4

Let $N = (S, T, W, M_{in})$ be a Petri net and let $s \in S$. Then $r\langle s \rangle$ is a region of $nl(N)$.

Proof.

Suppose $[\rho] \xrightarrow{u}_{nl(N)} [\rho']$. Then $r\langle s \rangle([\rho]) = M_\rho(s) \geq \sum_{t \in T} u(t) \cdot W(s,t) = \sum_{t \in T} u(t) \cdot {}^{r\langle s \rangle}t$ and $r\langle s \rangle([\rho']) = M_{\rho'}(s) = M_\rho(s) + \sum_{t \in T} u(t) \cdot (W(t,s) - W(s,t)) = r\langle s \rangle([\rho]) + \sum_{t \in T}(t^{r\langle s \rangle} - {}^{r\langle s \rangle}t)$. □

Using the above lemma we can prove the following result.

Lemma 4.4.5
Every PN-trace language satisfies the axioms (PL0) through (PL3).

Proof.
By Lemma 4.4.3 it is sufficient to prove that the L-trace languages associated with Petri nets satisfy these four axioms.

So let $N = (S, T, W, M_{in})$ be a Petri net. From the definition of MFS and I_N it follows directly that $nl(N)$ satisfies (PL0), (PL1), and (PL2).

In order to prove that (MFS, T, I_N) satisfies (PL3), suppose $\rho \in MFS$ and $\rho u \notin MFS$. Then by the definition of MFS there exists $s \in S$ such that $M_\rho(s) < \sum_{t \in T} u(t) \cdot W(s,t)$. Consider the region $r\langle s \rangle$ defined above. Then $r\langle s \rangle([\rho]) = M_\rho(s) < \sum_{t \in T} u(t) \cdot W(s,t) = \sum_{t \in T} u(t) \cdot {}^{r\langle s \rangle}t$. This implies that $r\langle s \rangle$ is non-trivial. This proves that $nl(N)$ satisfies (PL3). □

To prove that every L-trace language which satisfies the axioms (PL0) through (PL3) is a PN-trace language, a map ln from such L-trace languages to Petri nets is defined, using the map tn from multiset transition systems to Petri nets defined in Section 3.2.

Definition 4.4.6
Let $TL = (L, X, I)$ be an L-trace language satisfying (PL0) through (PL3). Then $ln(TL)$, the *Petri net associated with* TL, is the Petri net obtained by applying tn to the multiset transition system $(L/\cong, X, \longrightarrow_{TL}, [\underline{0}])$. Thus $ln(TL) = (\mathcal{R}_{TL}, X, W, M_{in})$ where

- $W : (\mathcal{R}_{TL} \times X) \cup (X \times \mathcal{R}_{TL}) \to \mathbf{N}$ is such that

$$\forall r \in \mathcal{R}_{TL}. \forall a \in X. (W(r,a) = {}^r a \text{ and } W(a,r) = a^r)$$

- $M_{in} : \mathcal{R}_{TL} \to \mathbf{N}$ is such that

$$\forall r \in \mathcal{R}_{TL}. M_{in}(r) = r([\underline{0}]).$$

□

Example 4.4.7
Consider the Petri net N_1 and its L-trace behaviour depicted in Figure 4.7. The Petri net $ln(nl(N_1))$ is depicted in Figure 4.9 with only some of its infinite number of places. The places of $ln(nl(N_1))$ are the non-trivial regions of $nl(N_1)$. Observe that the regions of $nl(N_1)$ do not necessarily satisfy the condition ${}^r a + {}^r c = a^r + c^r$. This contrasts with the situation in $nt(N_1)$, the marking diagram of N_1 which is depicted in Figure 3.5. The regions r_1, r_2, r_3, etc. are examples of places which do not have a counterpart in the Petri net $tn(nt(N_1))$ depicted in Figure 3.5, associated with the marking diagram of N_1. □

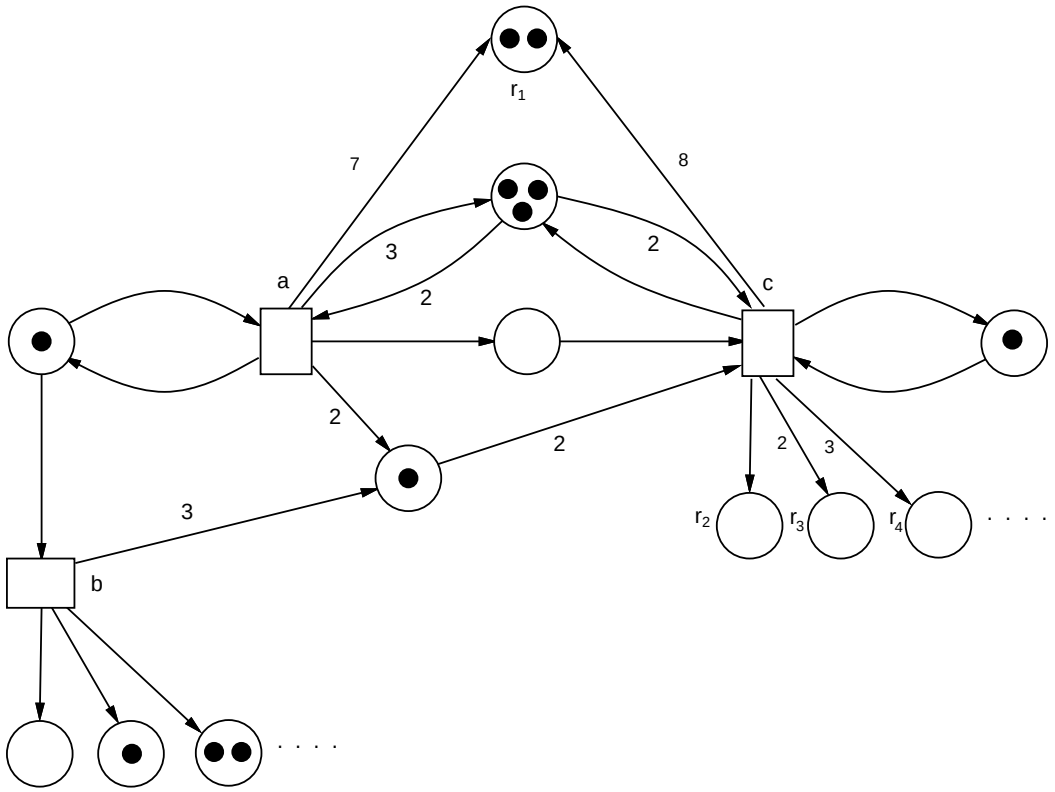


Figure 4.9: The Petri net $ln(nl(N_1))$

As an intermediate step in the proof that an L-trace language $TL = (L, X, I)$ satisfying the axioms (PL0) through (PL3) is LTL-isomorphic to the L-trace language generated by the Petri net $ln(TL)$, we prove separately that $L = MFS_{ln(TL)}$. First however we show that for every place s of $ln(TL)$, i.e. a region of TL, the region $r\langle s\rangle$ of $nl(ln(TL))$ associated with s in the way described in Lemma 4.4.4 and the region s of TL agree on all elements that their domains have in common. Thus, after having shown that $L = MFS_{ln(TL)}$, it follows that the non-trivial regions of $nl(ln(TL))$ are precisely the places of $ln(TL)$.

Lemma 4.4.8

Let $TL = (L, X, I)$ be an L-trace language satisfying (PL0) through (PL3) with $ln(TL) = (\mathcal{R}_{TL}, X, W, M_{in})$ and let $s \in \mathcal{R}_{TL}$. Then

$$\forall a \in X.\, r\langle s\rangle(a) = s(a) \text{ and}$$

$$\forall \rho \in (L \cap MFS_{ln(TL)}).\, r\langle s\rangle([\rho]_{ln(TL)}) = s([\rho]_I).$$

Proof.

By the definition of $r\langle s\rangle$ we have that for each $a \in X$, $r\langle s\rangle(a) = (W(s,a), W(a,s))$, and this equals $s(a)$ according to the definition of $ln(TL)$. Similarly, $r\langle s\rangle([\underline{0}]_{ln(TL)}) =$

$M_{in}(s)$ by the definition of $r\langle s\rangle$ and $s([\underline{0}]_I) = M_{in}(s)$ by the definition of $ln(TL)$. Note that $\underline{0} \in L \cap MFS_{ln(TL)}$. Now let us assume that $r\langle s\rangle([\rho]_{ln(TL)}) = s([\rho]_I)$ for all $\rho \in L \cap MFS_{ln(TL)}$ with $|\rho| < n$ where $n \geq 1$. Let $\rho \in L \cap MFS_{ln(TL)}$ be such that $\rho = \rho' u$ where $|\rho'| < n$ and $|\rho' u| \geq n$. Note that $[\rho']_{ln(TL)} \xrightarrow{u}_{nl(ln(TL))} [\rho]_{ln(TL)}$ and $[\rho']_I \xrightarrow{u}_{TL} [\rho]_I$. Then $r\langle s\rangle([\rho]_{ln(TL)}) = M_\rho(s) = M_{in}(s) + \sum_{a \in X} num_a(\rho) \cdot (W(a,s) - W(s,a))$. Thus $r\langle s\rangle([\rho]_{ln(TL)}) = M_{in}(s) + \sum_{a \in X} num_a(\rho') \cdot (W(a,s) - W(s,a)) + \sum_{a \in X} u(a) \cdot (W(a,s) - W(s,a)) = r\langle s\rangle([\rho']_{ln(TL)}) + \sum_{a \in X} u(a) \cdot (W(a,s) - W(s,a)) = s([\rho']_I) + \sum_{a \in X} u(a) \cdot (W(a,s) - W(s,a))$ by the induction hypothesis and this in turn equals $s([\rho']_I) + \sum_{a \in X} u(a) \cdot (a^s - {}^s a) = s([\rho]_I)$ because s is a region of L. Consequently $r\langle s\rangle(a) = s(a)$ for all $a \in X$ and $r\langle s\rangle([\rho]_{ln(TL)}) = s([\rho]_I)$ for all $\rho \in L \cap MFS_{ln(TL)}$. □

Using the correspondence described in this lemma between the regions of the original L-trace language and the regions of the L-trace language generated by the Petri net associated with this L-trace language, we can prove the following result.

Lemma 4.4.9

Let $TL = (L, X, I)$ be an L-trace language which satisfies the axioms (PL0) through (PL3). Then $L = MFS_{ln(TL)}$.

Proof.

In order to prove that $L \subseteq MFS_{ln(TL)}$, let $\rho \in L$. Then it easily follows from (PL1) that $[\underline{0}]_I \xrightarrow{\rho}_{TL}$. Hence we have by Lemma 3.2.7(2) that $\rho \in MFS_{ln(TL)}$.

In order to prove that $MFS_{ln(TL)} \subseteq L$, let $\rho \in MFS_{ln(TL)}$. If $\rho = \underline{0}$, then $\rho \in L$ by the non-emptiness and prefix-closedness of L. Now assume that $\rho = \rho' u$ with $u \neq \underline{0}$. Using an inductive argument we assume that $\rho' \in L$. Moreover, by Lemma 4.4.8, $s(\rho') = r\langle s\rangle(\rho') = M_{\rho'}(s)$ for all $s \in \mathcal{R}_{TL}$. Now assume to the contrary that $\rho = \rho' u \notin L$. Then by (PL3), there exists $r \in \mathcal{R}_{TL}$ such that $r(\rho') < \sum_{a \in X} u(a) \cdot {}^r a$. This would imply that $M_{\rho'}(r) < \sum_{a \in X} u(a) \cdot W(r,a)$. But this contradicts the fact that $\rho' u \in MFS_{ln(TL)}$. Hence $\rho' u \in L$. □

We now have that the underlying language of the original L-trace language and the set of multiset firing sequences of the Petri net associated with this L-trace language are equal. In order to prove that the original L-trace language and the L-trace language generated by this Petri net are isomorphic, the only thing left to prove is that concurrency is preserved.

Lemma 4.4.10

Let $TL = (L, X, I)$ be an L-trace language which satisfies the axioms (PL0) through (PL3). Then $TL \equiv nl(ln(TL))$.

Proof.

By Lemma 4.4.9, $L = MFS_{ln(TL)}$. Moreover, by (PL2), $I = \{(\rho, u) \mid \rho u \in L\} = \{(\rho, u) \mid \rho u \in MFS_{ln(TL)}\} = I_{ln(TL)}$. Hence id_{TL} is an LTL-isomorphism from TL to $nl(ln(TL))$. □

From Lemmas 4.4.5 and 4.4.10 we obtain the following result.

Theorem 4.4.11
An L-trace language is a PN-trace language iff it satisfies the axioms (PL0) through (PL3).

4.5 A Co-reflection Between $\mathcal{PTL}$ and $\mathcal{PN}$

The relationship between Petri nets and PN-trace languages established in the previous section can also be expressed in a categorical framework. We show in this section how the maps ln and nl can be extended to functors forming a co-reflection.

Definition 4.5.1
Let $\mathcal{PTL}$ be the category which has PN-trace languages as its objects and LTL-morphisms as its arrows. The identity morphism associated with an object TL is id_{TL}; composition of LTL-morphisms is composition of partial functions. □

The map nl can be extended to a functor from $\mathcal{PN}$ to $\mathcal{PTL}$ as follows.
Let N_i, $i = 1, 2$, be a pair of Petri nets and let (β, η) be a PN-morphism from N_1 to N_2. Then define $nl((\beta, \eta)) = \eta$.
Then the following result follows immediately from Lemma 2.2.2.

Lemma 4.5.2
nl is a functor from $\mathcal{PN}$ to $\mathcal{PTL}$. □

Now we extend the map ln to a functor from $\mathcal{PTL}$ to $\mathcal{PN}$. Recall that applying ln amounts to viewing first a PN-trace language $TL = (L, X, I)$ as a multiset transition system $lt(TL) = (L/\cong, X, \longrightarrow_{TL}, [\underline{0}])$, and then applying the map tn from Section 3.2 which yields the Petri net $ln(TL) = tn(lt(TL))$. When extending ln to the arrows of $\mathcal{PTL}$ we follow the same route: first lt is extended to the arrows of $\mathcal{PTL}$; next we use the definition of tn on MTS-morphisms as given in Section 3.2.

So let $TL_i = (L_i, X_i, I_i)$, $i = 1, 2$, be a pair of PN-trace languages with $lt(TL_i) = (L_i/\cong_{I_i}, X_i, \longrightarrow_{TL_i}, [\underline{0}]_{I_i})$, and let f be an LTL-morphism from TL_1 to TL_2. Then define $lt(f) = (f, g)$ where $g : L_1/\cong_{I_1} \to L_2/\cong_{I_2}$ is given by $g([\rho]_{I_1}) = [f(\rho)]_{I_2}$. Note that g is well-defined by Lemma 4.2.5. From Lemma 4.2.5 it also easily follows that $lt(f)$ is an MTS-morphism from $lt(TL_1)$ to $lt(TL_2)$. Hence by Lemma 3.2.9 there exists for each region r of TL_2 an inverse region $lt(f)^{-1}(r)$ of TL_1. In what follows we write $f^{-1}(r)$ instead of $lt(f)^{-1}(r)$. Thus $f^{-1}(r) : L_1/\cong_{I_1} \cup X_1 \to \mathbf{N} \cup (\mathbf{N} \times \mathbf{N})$ is given by:

(1) $\forall [\rho]_{I_1} \in L_1/\cong_{I_1} \,.\, f^{-1}(r)([\rho]_{I_1}) = r([f(\rho)]_{I_2})$

(2) $\forall a \in X_1.\, f^{-1}(r)(a) = \begin{cases} r(f(a)) & \text{if } f(a) \text{ is defined} \\ (0,0) & \text{otherwise.} \end{cases}$

Now define $ln(f) = tn(lt(f))$. Thus $ln(f) = (\beta_f, \eta_f)$ where $\eta_f = f$ and $\beta_f : \mathcal{R}_{TL_2} \to \mathcal{R}_{TL_1}$ is given by:

$$\beta_f(r) = \begin{cases} f^{-1}(r) & \text{if } f^{-1}(r) \text{ is non-trivial} \\ \text{undefined} & \text{otherwise.} \end{cases}$$

Lemma 4.5.3

ln is a functor from $\mathcal{PTL}$ to $\mathcal{PN}$.

Proof.

By Lemma 3.2.10, whenever TL_1 and TL_2 are PN-trace languages, and f is an LTL-morphism from TL_1 to TL_2, then $ln(f)$ is a PN-morphism from $ln(TL_1)$ to $ln(TL_2)$. Hence it is sufficient to prove that *ln* preserves identities and respects composition. Clearly *ln* preserves identities. Now assume that f_1 is an LTL-morphism from TL_1 to TL_2 with $ln(f_1) = (\beta_{f_1}, \eta_{f_1})$ and f_2 is an LTL-morphism from TL_2 to TL_3 with $ln(f_2) = (\beta_{f_2}, \eta_{f_2})$. Suppose $ln(f_2 \circ f_1) = (\beta_{f_2 \circ f_1}, \eta_{f_2 \circ f_1})$. Then $\eta_{f_2 \circ f_1} = f_2 \circ f_1 = \eta_{f_2} \circ \eta_{f_1}$. Because $ln(TL)$ is S-simple and has no isolated places we have by Lemma 2.2.4 that $ln(f_2 \circ f_1) = (\beta_{f_2 \circ f_1}, \eta_{f_2 \circ f_1}) = (\beta_{f_1} \circ \beta_{f_2}, \eta_{f_2} \circ \eta_{f_1}) = (\beta_{f_2}, \eta_{f_2}) \circ (\beta_{f_1}, \eta_{f_1}) = ln(f_2) \circ ln(f_1)$. □

Next we prove the main result of this section stating that *ln* and *nl* form a co-reflection with *ln* as the left adjoint.

We first define the PN-morphisms which turn out to form the co-unit of the adjunction.

Given a Petri net $N = (S, T, W, M_{in})$ with $nl(N) = (MFS, T, I_N)$ and $ln(nl(N)) = (\mathcal{R}_{nl(N)}, T, \hat{W}, \hat{M}_{in})$, let $\epsilon_S : S \to \mathcal{R}_{nl(N)}$ be defined by:

$$\epsilon_S(s) = \begin{cases} r\langle s \rangle & \text{if } r\langle s \rangle \text{ is non-trivial} \\ \text{undefined} & \text{otherwise.} \end{cases}$$

Furthermore, define $\epsilon_T : T \to T$ by: $\epsilon_T(t) = t$. Note that ϵ_S is well-defined by Lemma 4.4.4.

Lemma 4.5.4

Let $N = (S, T, W, M_{in})$ be a Petri net with $nl(N) = (MFS, T, I_N)$ and $ln(nl(N)) = (\mathcal{R}_{nl(N)}, T, \hat{W}, \hat{M}_{in})$. Then (ϵ_S, ϵ_T) is a PN-morphism from $ln(nl(N))$ to N.

Proof.

Suppose $s \in S$ is such that $\epsilon_S(s)$ is defined. Then $\hat{M}_{in}(\epsilon_S(s)) = \hat{M}_{in}(r\langle s \rangle) = r\langle s \rangle([\underline{0}]_N) = M_{in}(s)$ which proves condition (1) in the definition of PN-morphism.

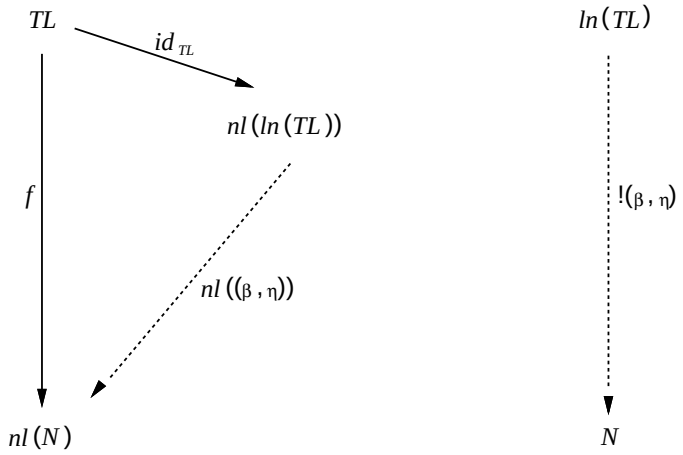
Because ϵ_T is a total function, condition (2) in the definition of PN-morphism trivially holds.

In order to prove condition (3), suppose $t \in T$. If $s \in \epsilon_S^{-1}({}^\bullet t)$ then we must have that $r\langle s \rangle \in {}^\bullet t$, that is $\hat{W}(r\langle s \rangle, t) > 0$. This implies that ${}^{r\langle s \rangle}t > 0$ and hence $W(s,t) > 0$. This proves that $s \in {}^\bullet t = {}^\bullet \epsilon_T(t)$. On the other hand, if $s \in {}^\bullet \epsilon_T(t) = {}^\bullet t$, then ${}^{r\langle s \rangle}t = W(s,t) > 0$. This implies that $r\langle s \rangle$ is non-trivial and $\hat{W}(r\langle s \rangle, t) = {}^{r\langle s \rangle}t > 0$. Then $r\langle s \rangle \in {}^\bullet t$ and hence $s \in \epsilon_S^{-1}({}^\bullet t)$. Moreover, $W(s, \epsilon_T(t)) = W(s,t) = \hat{W}(r\langle s \rangle, t) = \hat{W}(\epsilon_S(s), t)$. Similarly it can be proved that $\epsilon_S^{-1}(t^\bullet) = \epsilon_T(t)^\bullet$ and $W(\epsilon_T(t), s) = \hat{W}(t, \epsilon_S(s))$. This proves condition (3) in the definition of PN-morphism. □

Now we can prove the main result of this section.

Theorem 4.5.5
$ln : \mathcal{PTL} \to \mathcal{PN}$ and $nl : \mathcal{PN} \to \mathcal{PTL}$ form a co-reflection with ln the left adjoint and the arrows id_{TL} as unit.

Proof.
Let $TL = (L, X, I)$ be an PN-trace language, let $N = (S, T, W, M_{in})$ be a Petri net, and let f be an LTL-morphism from TL to $nl(N) = (MFS_N, T, I_N)$. We must show that there is a unique PN-morphism (β, η) from $ln(TL) = (\mathcal{R}_{TL}, X, W_{TL}, M)$ to N such that the following diagram commutes.



Define (β, η) by $(\beta, \eta) = (\epsilon_S, \epsilon_T) \circ ln(f)$. Hence $\beta : S \to \mathcal{R}_{TL}$ is such that for all $s \in S$, $\beta(s) = f^{-1}(r\langle s \rangle)$ if $f^{-1}(r\langle s \rangle)$ is non-trivial and $\beta(s)$ is undefined otherwise. The function $\eta : X \to T$ is such that $f = \eta$. Because (ϵ_S, ϵ_T) and $ln(f)$ are PN-morphisms by Lemma 4.5.4 and Lemma 4.5.3 respectively, and because the composition of PN-morphisms is again a PN-morphism, the pair (β, η) is a PN-morphism.

Since $nl((\beta, \eta)) = \eta = f$, it is clear that the diagram commutes. Moreover, (β, η) is the unique PN-morphism for which the diagram commutes, because if (β', η') is such that the diagram commutes, then $\eta = \eta'$ by the definition of nl, and hence also $\beta = \beta'$ by Lemma 2.2.4. □

From this proof it easily follows that the PN-morphisms (ϵ_S, ϵ_T) defined above form the co-unit of this adjunction.

4.6 L-Trace Languages and M-trace Languages

Based on the intuition underlying L-traces, there is a natural way to view M-trace languages as L-trace languages. In this section we show that this leads to a bijection between the class of reduced M-trace languages and a certain subclass of L-trace languages. It turns out that for 1-safe Petri nets the trace semantics in terms of M-trace languages agrees with the trace semantics in terms of L-trace languages via this bijection. Finally we show that this bijection between M-trace languages and

a subclass of L-trace languages can be extended to their corresponding categories, leading to a categorical equivalence.

Given an M-concurrency alphabet (X, Ind), let

$$I_{Ind} = \{(\rho, u) \mid \rho \in (M_F(X))^+ \text{ and } u \in P_F(X) \text{ and } \forall a, b \in u.\, (a \neq b \Rightarrow (a, b) \in Ind)\}$$

be the *L-independence relation associated with* (X, Ind).

So in I_{Ind} a set of symbols is independent iff all symbols in the set are pairwise independent in Ind. Moreover, such sets are independent in every possible context, thus reflecting the globality of the M-independence relation Ind. Thus for each M-concurrency alphabet (X, Ind) with M-equivalence $\simeq_{Ind}$ given through $\doteq_{Ind}$, we obtain an L-concurrency alphabet (X, I_{Ind}) and hence an L-equivalence relation $\cong_{I_{Ind}}$ through $\doteq_{I_{Ind}}$.

In what follows we write $\rho \mathrel{\hat{\doteq}_{Ind}} \rho'$ and $\rho \hat{\cong}_{Ind} \rho'$ rather than $\rho \doteq_{I_{Ind}} \rho'$ and $\rho \cong_{I_{Ind}} \rho'$, respectively. If Ind is clear from the context then we may omit the subscript $_{Ind}$ in $\mathrel{\hat{\doteq}_{Ind}}$ and $\hat{\simeq}_{Ind}$.

Clearly, an alternative characterization of $\mathrel{\hat{\doteq}}$ is given by:
$\forall \rho, \rho' \in (M_F(X))^+.\, \rho \mathrel{\hat{\doteq}} \rho' \Leftrightarrow \exists \rho_1, \rho_2 \in (M_F(X))^+, u, v, u', v' \in P_F(X).$

$[(1)\ \rho = \rho_1 u v \rho_2$ and $\rho' = \rho_1 u' v' \rho_2$ and

$(2)\ u \cup v = u' \cup v'$ and $u \cap v = u' \cap v' = \emptyset$

$(3)\ \forall a, b \in u \cup v.\, (a \neq b \Rightarrow (a, b) \in Ind)]$. □

By the following lemma I_{Ind} agrees with Ind when restricted to ordinary sequences. (Recall that we view ordinary sequences as step/multiset sequences by the monoid homomorhism which maps each action a to the singleton containing a).

Lemma 4.6.1

Let (X, Ind) be an M-concurrency alphabet and let $\rho, \rho' \in X^*$. Then

$$\rho \simeq_{Ind} \rho' \Leftrightarrow \rho \hat{\simeq}_{Ind} \rho'.$$

Proof.

Since $\rho \doteq_{Ind} \rho'$ implies that $\rho \mathrel{\hat{\doteq}_{Ind}} \rho'$, we also have that $\rho \simeq_{Ind} \rho'$ implies that $\rho \hat{\simeq}_{Ind} \rho'$. The implication in the other direction follows easily from Lemma 4.1.3. □

Each M-trace language (L, X, Ind) leads to an L-trace language which has as its underlying language the set $\hat{L}$ of those multiset sequences which are equivalent (under $\hat{\simeq}_{Ind}$) to some element from L. Note that $\hat{L} \subseteq (P_F(X))^+$. Since we prefer to work with reduced systems, we choose for the underlying L-independence relation of this L-trace language however not I_{Ind}, but the least L-independence relation guaranteeing that steps in $\hat{L}$ are independent. Consequently, information about independencies in Ind which are not "used" in $\hat{L}$ is lost.

So given an M-trace language $TL = (L, X, Ind)$, define $ml(TL) = (\hat{L}, X, \hat{I})$ where

- $\hat{L} = \bigcup\{[\rho]_{I_{Ind}} \mid \rho \in L\}$

- $\hat{I} = \{(\rho, u) \mid \rho u \rho' \in \hat{L}\}$.

Since $L \subseteq X^* \subseteq (P_F(X))^+$, it is easy to see that also $\hat{L} \subseteq (P_F(X))^+$ and $\hat{I} \subseteq (P_F(X))^+ \times P_F(X)$.

From the definition of $\hat{L}$, it is immediately clear that $\hat{L}$ is consistent with respect to (X, I_{Ind}). As the following lemma shows, $\hat{L}$ is also consistent with respect to $(X, \hat{I})$ so that $ml(TL)$ is indeed an L-trace language.

Lemma 4.6.2

Let $TL = (L, X, Ind)$ be an M-trace language. Then $ml(TL) = (\hat{L}, X, \hat{I})$ is an L-trace language.

Proof.

It must be proved that $\hat{L}$ is consistent with respect to $(X, \hat{I})$. Suppose $\rho \doteq_{\hat{I}} \rho'$ and $\rho \in \hat{L} \subseteq (P_F(X))^+$. Then there exist $\rho_1, \rho_2 \in (P_F(X))^+$ and $u, v, u', v' \in P_F(X)$ such that $\rho = \rho_1 u v \rho_2$, $\rho' = \rho_1 u' v' \rho_2$, $u \cap v = u' \cap v' = \emptyset$, $u \cup v = u' \cup v'$, and $(\rho_1, u \cup v) \in \hat{I}$. By the definition of $\hat{I}$ there exists $\rho_1(u \cup v)\rho'' \in \hat{L}$. Since $\rho_1(u \cup v)\rho'' \hat{\simeq}_{Ind} \rho'$ for some $\rho' \in L \subseteq X^*$, this implies that $(a, b) \in Ind$ for all $a, b \in u \cup v$ with $a \neq b$. Hence $\rho \hat{\doteq}_{Ind} \rho'$. Thus $\rho' \in \hat{L}$ because $\rho \in \hat{L}$. This proves that $\hat{L}$ is consistent with respect to $(X, \hat{I})$. □

When restricted to reduced M-trace languages, the map ml is injective. In fact, a class of *ML-trace languages* can be identified, which through ml is in bijective correspondence with the class of reduced M-trace languages.

Definition 4.6.3

An L-trace language $TL = (L, X, I)$ is an *ML-trace language* if it satisfies the following three conditions.

(ML1) $(\rho, u) \in I \Leftrightarrow \exists \rho' \in (P_F(X))^+ . \rho u \rho' \in L$

(ML2) $\rho u \rho' \in L \Rightarrow \forall a \in X . u(a) \leq 1$

(ML3) $(\rho_1 u v \rho_2 \in L$ and $u, v \in P_F(X)$ and $u \cap v = \emptyset$ and $\exists \rho \in L . \forall a, b \in u \cup v . (\rho, \{a, b\}) \in I) \Rightarrow \rho_1(u \cup v)\rho_2 \in L$. □

Condition (ML1) states that the L-independence relation of TL agrees with the underlying language of TL. Condition (ML2) forbids "auto-concurrency" in TL. Finally, condition (ML3) captures both globality of concurrency and the fact that arbitrary concurrency can be derived from binary concurrency.

Note that in the presence of (ML2), the condition $u, v \in P_F(X)$ from (ML3) is already satisfied. Also note that (ML1) and (ML2) together imply that $I \subseteq (P_F(X))^+ \times P_F(X)$.

Lemma 4.6.4

Let $TL = (L, X, Ind)$ be an M-trace language. Then $ml(TL) = (\hat{L}, X, \hat{I})$ is an ML-trace language.

Proof.

By Lemma 4.6.2, $ml(TL)$ is an L-trace language.

It is clear that $ml(TL)$ satisfies (ML1) and (ML2). Now suppose $\rho_1 uv\rho_2 \in \hat{L}$ is such that $u \cap v = \emptyset$ and, for all $a, b \in u \cup v$, there exists $\rho\{a,b\}\rho' \in \hat{L}$. Then for all such $\rho\{a,b\}\rho' \in \hat{L}$, $\rho\{a,b\}\rho' \hat{\cong}_{Ind}\ \rho''$ for some $\rho'' \in L$ by the definition of $\hat{L}$. This implies that $(a,b) \in Ind$ for all $a, b \in u \cup v$ with $a \neq b$ and hence $\rho_1 uv\rho_2 \hat{\doteq}_{Ind}\ \rho_1(u \cup v)\rho_2$. We can now conclude that $\rho_1(u \cup v)\rho_2 \in \hat{L}$. □

Now we define the map lm from the class of ML-trace languages to the class of reduced M-trace languages which turns out to be the inverse of ml.

Given an ML-trace language $TL = (L, X, I)$, define $lm(TL) = (L_{TL}, X, Ind_{TL})$ where

- $L_{TL} = L \cap X^*$
- $Ind_{TL} = \{(a,b) \mid a \neq b \text{ and } \exists(\rho, \{a,b\}) \in I\}$.

Lemma 4.6.5

Let $TL = (L, X, I)$ be an ML-trace language. Then $lm(TL) = (L_{TL}, X, Ind_{TL})$ is a reduced M-trace language.

Proof.

Suppose $\rho_1 ab\rho_2 \doteq_{Ind_{TL}} \rho_1 ba\rho_2$ where $\rho_1 ab\rho_2 \in L_{TL} \subseteq L$ and $(a,b) \in Ind_{TL}$. In order to prove that $lm(TL)$ is an M-trace language, it is sufficient to prove that then also $\rho_1 ba\rho_2 \in L_{TL}$. By the definition of Ind_{TL}, $a \neq b$ and there exists $(\rho, \{a,b\}) \in I$. Then by (ML1) there exists $\rho\{a,b\}\rho' \in L$. From (ML3) it then follows that also $\rho_1\{a,b\}\rho_2 \in L$ and hence $(\rho_1, \{a,b\}) \in I$ by (ML1). Then $\rho_1 ab\rho_2 \doteq_I \rho_1 ba\rho_2$, and so $\rho_1 ba\rho_2 \in L$ by the consistency of L. Moreover, $\rho_1 ba\rho_2 \in L \cap X^*$ because $\rho_1 ab\rho_2 \in X^*$. This proves that $\rho_1 ba\rho_2 \in L_{TL}$.

In order to prove that $lm(TL)$ is reduced, suppose $(a,b) \in Ind_{TL}$. Then $a \neq b$ and there exists $(\rho, \{a,b\}) \in I$. This implies that there exists $\rho\{a,b\}\rho' \in L$ by (ML1). Now it easily follows from (ML1) that $\rho\{a,b\}\rho' \cong_I \rho_1 ab\rho_1'$ for some $\rho_1 ab\rho_1' \in L \cap X^* = L_{TL}$. □

As the following lemma shows, the L-independence relation of an ML-trace language TL agrees with the M-independence relation of $lm(TL)$.

Lemma 4.6.6

Let $TL = (L, X, I)$ be an ML-trace language with $lm(TL) = (L_{TL}, X, Ind_{TL})$ and let $\rho \in L$ and $\rho' \in (P_F(X))^+$. Then

$$\rho \doteq_I \rho' \Leftrightarrow \rho\ \hat{\doteq}_{Ind_{TL}}\ \rho'.$$

Proof.

Suppose $\rho \doteq_I \rho'$ where $\rho = \rho_1 uv\rho_2$, $\rho' = \rho_1 u'v'\rho_2$, $u \cup v = u' \cup v'$, $u \cap v = u' \cap v' = \emptyset$, and $(\rho_1, u \cup v) \in I$. Then for all $a, b \in u \cup v$ with $a \neq b$, $\rho \doteq_I \rho_1\{a,b\}(u \cup v - \{a,b\})\rho_2$ and hence also $\rho_1\{a,b\}(u \cup v - \{a,b\})\rho_2 \in L$. This implies that for all $a, b \in u \cup v$ with $a \neq b$, $(\rho_1, \{a,b\}) \in I$ by (ML1), and hence $(a,b) \in Ind_{TL}$. Thus $\rho\ \hat{\doteq}_{Ind_{TL}}\ \rho'$.

Now suppose $\rho \hat{\doteq}_{Ind_{TL}} \rho'$ where $\rho = \rho_1 uv\rho_2$, $\rho' = \rho_1 u'v'\rho_2$, $u \cup v = u' \cup v'$, $u \cap v = u' \cap v' = \emptyset$, and, for all $a, b \in u \cup v$ with $a \neq b$, $(a, b) \in Ind_{TL}$. Then by the definition of Ind_{TL} and (ML1) there exist $\rho_1'\{a, b\}\rho_2' \in L$ for all $a, b \in u \cup v$ with $a \neq b$. Then $\rho_1(u \cup v)\rho_2 \in L$ by (ML3), and hence $(\rho_1, u \cup v) \in I$ by (ML1). This implies that $\rho \doteq_I \rho'$. □

Using this lemma we can now prove the desired bijection.

Theorem 4.6.7

ml is a bijection from the class of reduced M-trace languages to the class of ML-trace languages, with inverse *lm*.

Proof.

In order to prove that *ml* is injective, suppose $TL_i = (L_i, X_i, Ind_i)$, $i = 1, 2$, are reduced M-trace languages such that $ml(TL_1) = ml(TL_2) = (L, X, I)$. Then $X_1 = X_2 = X$ by the definition of *ml*. Moreover, $L = \bigcup\{[\rho]_{I_{Ind_1}} \mid \rho \in L_1\} = \bigcup\{[\rho]_{I_{Ind_2}} \mid \rho \in L_2\}$. In order to prove that $L_1 \subseteq L_2$, let $\rho \in L_1 \subseteq L$. Then $\rho \hat{\simeq}_{Ind_2} \rho'$ for some $\rho' \in L_2$. Hence also $\rho \simeq_{Ind_2} \rho'$ by Lemma 4.6.1. Thus $\rho \in L_2$ by the consistency of L_2. Similarly it can be proved that $L_2 \subseteq L_1$ and hence $L_1 = L_2$.

In order to prove that $Ind_1 = Ind_2$, suppose $(a, b) \in Ind_1$. Then there exists $\rho ab\rho' \in L_1$ because TL_1 is reduced. Moreover, $\rho ab\rho' \hat{\doteq}_{Ind_1} \rho\{a, b\}\rho'$ and hence $\rho\{a, b\}\rho' \in L$. This implies that $\rho\{a, b\}\rho' \hat{\simeq}_{Ind_2} \rho''$ for some $\rho'' \in L_2$. It now easily follows that $(a, b) \in Ind_2$. This proves that $Ind_1 \subseteq Ind_2$. Similarly we have that $Ind_2 \subseteq Ind_1$ and hence $Ind_1 = Ind_2$. We can conclude that *ml* is injective.

Suppose $TL = (L, X, I)$ is an ML-trace language with $lm(TL) = (L_{TL}, X, Ind_{TL})$ and $ml(lm(TL)) = (\hat{L}, X, \hat{I})$. It must be proved that $L = \hat{L}$ and $I = \hat{I}$.

Suppose $\rho \in L$. Then it easily follows from (ML1) that there exists $\rho' \in L \cap X^* = L_{TL}$ with $\rho \cong_I \rho'$. Hence $\rho \hat{\simeq}_{Ind_{TL}} \rho'$ by Lemma 4.6.6. Since $\rho' \in L_{TL}$, we then have that $\rho \in \hat{L}$ by the definition of *ml*. This proves that $L \subseteq \hat{L}$.

Now suppose $\rho \in \hat{L}$. Then $\rho \hat{\simeq}_{Ind_{TL}} \rho'$ for some $\rho' \in L_{TL} \subseteq \hat{L}$. Hence $\rho' \in L \cap X^*$ by the definition of *lm*. Then $\rho \cong_I \rho'$ by Lemma 4.6.6 and hence $\rho \in L$ by the consistency of L. This proves that $\hat{L} \subseteq L$ and hence $L = \hat{L}$. Then also $I = \hat{I}$ by (ML1). We can now conclude that *ml* is a bijection with inverse *lm*. □

Thus the class of reduced M-trace languages can be viewed as a subclass of the class of L-trace languages. Note that it would have been possible to define *ml* such that it is injective on the class of all M-trace languages, by taking I_{Ind} for the L-independence relation of $ml((L, X, Ind))$. The reason for restricting our attention to M-trace languages which are reduced is, that we want the bijection to be such that it maps the M-trace language $sm(N)$ associated with a 1-safe Petri net N to the L-trace language $nl(N)$ associated with N. Next we show that this goal is achieved with the present definition of *ml*.

First we need the following lemma.

Lemma 4.6.8

Let N be a 1-safe Petri net, let $\rho \in SFS$, and let $\rho' \in (P_F(T))^+$. Then

$$\rho \doteq_N \rho' \Leftrightarrow \rho \hat{\doteq}_{Ind_N} \rho'.$$

Proof.

First assume that $\rho \doteq_N \rho'$. Suppose $\rho = \rho_1 uv\rho_2$, $\rho' = \rho_1 u'v'\rho_2$, $u \cup v = u' \cup v'$, $u \cap v = u' \cap v' = \emptyset$, and $(\rho_1, u \cup v) \in I_N$. Then $\rho_1(u \cup v) \in SFS$ by the definition of I_N. Moreover, for all $t, t' \in u \cup v$ with $t \neq t'$, $M_{\rho_1}[\{t, t'\}\rangle$ and hence $(t, t') \in Ind_N$. This implies that also $\rho \stackrel{\cdot}{\doteq}_{Ind_N} \rho'$.

Now assume that $\rho \stackrel{\cdot}{\doteq}_{Ind_N} \rho'$ and suppose $\rho = \rho_1 uv\rho_2$, $\rho' = \rho_1 u'v'\rho_2$, $u \cup v = u' \cup v'$, $u \cap v = u' \cap v' = \emptyset$, and for all $t, t' \in u \cup v$ with $t \neq t'$, $(t, t') \in Ind_N$. Then $M_{\rho_1} \in \mathcal{RM}$ because $\rho \in SFS$, and hence $M_{\rho_1}[u \cup v\rangle$ by Lemma 2.1.10. Thus $\rho_1(u \cup v) \in SFS$. This implies by the definition of I_N that $(\rho_1, u \cup v) \in I_N$. We can conclude that $\rho \doteq_N \rho'$. □

Now we can prove that the trace semantics for Petri nets in terms of L-traces extends the trace semantics for 1-safe Petri nets in terms of M-traces.

Theorem 4.6.9

Let $N = (S, T, W, M_{in})$ be a 1-safe Petri net with $sm(N) = (FS, T, Ind_N)$ and $nl(N) = (MFS, T, I_N)$. Then $ml(sm(N)) \equiv nl(N)$.

Proof.

First note that $MFS = SFS$ because N is 1-safe. Let $ml(sm(N)) = (L, T, I)$. In order to prove that $L \subseteq SFS$, let $\rho \in L$. Then $\rho \stackrel{\cdot}{\cong}_{Ind_N} \rho'$ for some $\rho' \in FS \subseteq SFS$. By Lemma 4.6.8 also $\rho \cong_N \rho'$. This then implies by Lemma 4.3.2 that $\rho \in SFS$. This proves that $L \subseteq SFS$.

Now suppose $\rho \in SFS$. Then there exists $\rho' \in FS \subseteq L$ such that $\rho \cong_N \rho'$. Then again by Lemma 4.6.8, also $\rho \stackrel{\cdot}{\cong}_{Ind_N} \rho'$. Hence $\rho \in L$. This proves that $L = SFS$. Then also $I_N = \{(\rho, u) \mid \rho u \in SFS\} = \{(\rho, u) \mid \rho u \in L\} = I$. Hence $id_{nl(N)}$ is the required LTL-isomorphism. □

To conclude this section we prove that the bijection between ML-trace languages and reduced M-trace languages can be lifted to a categorical equivalence.

Definition 4.6.10

Let $\mathcal{MTL}$ be the category which has reduced M-trace languages as its objects and MTL-morphisms as its arrows. The identity morphism associated with an object is the identity function on its alphabet and composition of arrows is composition of partial functions. □

Even though we only consider the category of reduced M-trace languages, every M-trace language can in fact be represented canonically as such a reduced M-trace language. This follows from the easy to prove observation that $\mathcal{MTL}$ is a full co-reflective subcategory of the category which has *all* M-trace languages as its objects.

Definition 4.6.11

Let $\mathcal{MLTL}$ be the category which has ML-trace languages as its objects and LTL-morphisms as its arrows. The identity morphism associated with an object TL is id_{TL}; composition of arrows is composition of partial functions. □

In order to extend the map ml to a functor, define $ml(f) = f$ for an MTL-morphism f between reduced M-trace languages.

Lemma 4.6.12

ml is a functor from $\mathcal{MTL}$ to $\mathcal{MLTL}$.

Proof.

Let $TL_i = (L_i, X_i, Ind_i)$, $i = 1, 2$, be reduced M-trace languages and let f be an MTL-morphism from TL_1 to TL_2. It is by Lemma 4.6.4 sufficient to prove that $ml(f) = f$ is an LTL-morphism from $ml(TL_1) = (\hat{L}_1, X_1, \hat{I}_1)$ to $ml(TL_2) = (\hat{L}_2, X_2, \hat{I}_2)$.

Suppose $\rho \in \hat{L}_1$. Then $\rho \,\hat{\simeq}_{Ind_1}\, \rho'$ for some $\rho' \in L_1$. Then because f is an MTL-morphism from TL_1 to TL_2, $f(\rho') \in L_2$. In order to prove that $f(\rho) \in \hat{L}_2$ it is then sufficient to prove that $f(\rho) \,\hat{\simeq}_{Ind_2}\, f(\rho')$. Suppose $\rho_1 u v \rho_2 \,\hat{\dot{-}}_{Ind_1}\, \rho_1 u' v' \rho_2$ with $u \cap v = u' \cap v' = \emptyset$, $u \cup v = u' \cup v'$, and $(a, b) \in Ind_1$ for all $a, b \in u \cup v$ with $a \neq b$. Then $(f(a), f(b)) \in Ind_2$ for all $a, b \in u \cup v$ with $a \neq b$ and $f(a)$ and $f(b)$ both defined because f is an MTL-morphism. Then also $f(a) \neq f(b)$ for all such $a, b \in u \cup v$ by the irreflexivity of Ind_2. Hence $f(\rho_1)f(u)f(v)f(\rho_2) \,\hat{\dot{-}}_{Ind_2}\, f(\rho_1)f(u')f(v')f(\rho_2)$. We can now conclude that $f(\rho) \,\hat{\simeq}_{Ind_2}\, f(\rho')$.

From the definition of ml it is clear that then also $f(I_1) \subseteq I_2$. This proves that f is an LTL-morphism from $ml(TL_1)$ to $ml(TL_2)$. □

In order to extend the map lm to a functor, also define $lm(f) = f$ for an LTL-morphism f between ML-trace languages.

Lemma 4.6.13

lm is a functor from $\mathcal{MLTL}$ to $\mathcal{MTL}$.

Proof.

Let $TL_i = (L_i, X_i, I_i)$, $i = 1, 2$, be ML-trace languages and let f be an LTL-morphism from TL_1 to TL_2. It is by Lemma 4.6.5 sufficient to prove that $lm(f)$ is an MTL-morphism from $lm(TL_1) = (L_{TL_1}, X_1, Ind_{TL_1})$ to $lm(TL_2) = (L_{TL_2}, X_2, Ind_{TL_2})$.

Suppose $\rho \in L_{TL_1} = L_1 \cap {X_1}^*$. Then $f(\rho) \in L_2$ because f is an LTL-morphism. Hence also $f(\rho) \in L_2 \cap {X_2}^* = L_{TL_2}$.

Now suppose $(a, b) \in Ind_{TL_1}$ is such that $f(a)$ and $f(b)$ are both defined. Then $a \neq b$ and, by the definition of Ind_{TL_1}, there exists $(\rho, \{a, b\}) \in I_1$. So $(f(\rho), \{f(a), f(b)\}) \in I_2$ because f is an LTL-morphism. Moreover, $f(a) \neq f(b)$ by (ML1) and (ML2). Hence $(f(a), f(b)) \in Ind_{TL_2}$ by the definition of Ind_{TL_2}. □

Now we have the following result.

Theorem 4.6.14

$\mathcal{MTL}$ and $\mathcal{MLTL}$ are equivalent.

Proof.

It is sufficient to prove that the functor ml is full and faithful, and that for every object TL of $\mathcal{MLTL}$ there exists an object TL' of $\mathcal{MTL}$ such that $ml(TL') \equiv TL$.

This last property follows immediately from Theorem 4.6.7. Moreover, from the definition of ml we immediately have that the functor ml is faithful. Finally, if f is an LTL-morphism in $\mathcal{MLTL}$ from TL to TL', then by Lemma 4.6.13 $lm(f) = f$ is an MTL-morphism from $lm(TL)$ to $lm(TL')$. Since $ml(lm(TL)) = TL$ and $ml(lm(TL')) = TL'$ by Theorem 4.6.7, this proves that ml is also full. Hence we can conclude that $\mathcal{MTL}$ and $\mathcal{MLTL}$ are equivalent. □

4.7 Concluding Remarks

The generalization of the classical trace model introduced in this chapter provides a tool for the description of the behaviour of Petri nets. What is missing at this stage is a sound theoretical basis for the theory of local traces.

In the last years there have been several other proposals for extending the theory of traces. Several of these, though differently motivated, focus on similar generalizations, in particular context-dependency or the use of step sequences (see, e.g., [58, 87, 83, 3, 91, 44]). At present it is however not clear if useful connections can be established between our approach and the cited related approaches.

For 1-safe Petri nets also *processes* [76] have been proposed to represent its runs, leading to a partial order description of the transition occurrences. Such a process is obtained by partially unfolding the Petri net (starting from the places which are initially marked) while resolving conflicts. Thus processes lead to special kinds of nets, called *causal nets*, which are acyclic and in which places have at most one input and one output transition. The transitions of a causal net represent the occurrences of transitions in the original Petri net and places represent the tokens in the original Petri net, where the set of minimal places (i.e. places with no input transitions) corresponds to the initial marking. It turns out that there is a one-to-one correspondence between the M-traces and the processes associated with a 1-safe Petri net (see [67] for the proof in the context of elementary net systems).

Also for general Petri nets causal nets can be used for defining processes [34]. Because now Petri nets are not necessarily 1-safe, this however leads to a distinction between tokens in a place.

In Figure 4.10 the Petri net N_6 from Figure 4.2 is depicted together with two processes which correspond to the observation of the occurrence of c followed by the occurrence of a. In the first process the transition a occurs where it consumes the token in place s_2 which was already there in the initial marking whereas in the second process the transition a consumes the token in place s_2 which has just been put there by transition c. It has been argued in [9] that such a distinction between tokens in a place is often undesirable. Therefore it has been suggested that a better representation of a run is given by an equivalence class of processes. The equivalence relation over processes is defined in such a way that if two places in the causal net are not causally related and represent tokens in the same place in the original Petri net, then this

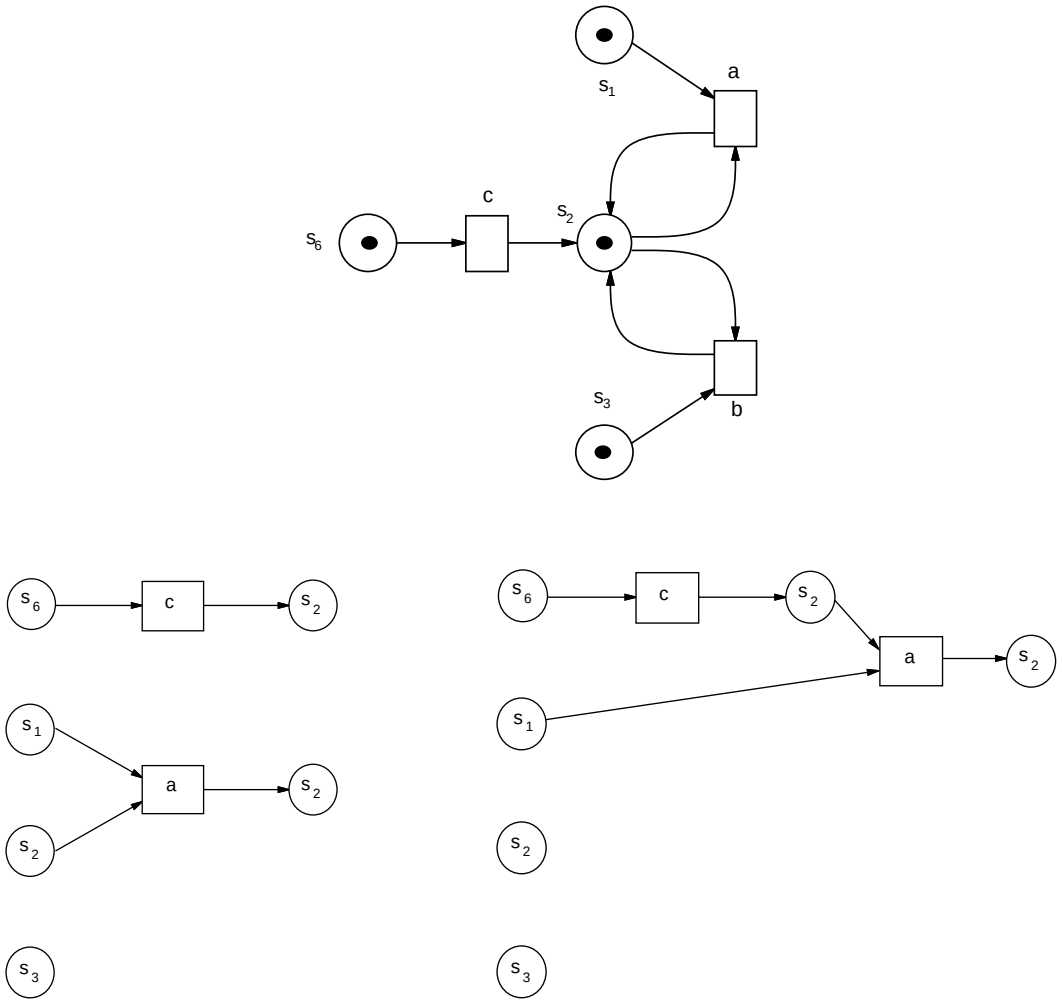


Figure 4.10: Two processes of N_6

process is equivalent to the process obtained by "swapping" the parts of the causal net "after" these places. For instance, in the first process in Figure 4.10 the input place labelled with s_2 of the transition labelled with a and the output place of the transition labelled with c are two places with the same label which are not causally related. The (equivalent) process obtained by swapping their consequences is the second process in Figure 4.10.

It has been shown in [9] that these equivalence classes of processes are in one-to-one correspondence with equivalence classes of occurrence sequences. This equivalence relation over occurrence sequences identifies occurrence sequences which only differ in the order of concurrent transitions.

Definition 4.7.1

Let $N = (S, T, W, M_{in})$ be a Petri net.

(1) $\sim_N$ is the least relation over occurrence sequences such that if $M_0 t_1 M_1 \ldots t_n M_n$

is an occurrence sequence with $n \geq 0$ and if $1 \leq i \leq n-1$ is such that

$$\forall s \in S.\, M_{i-1}(s) \geq W(s,t_i) + W(s,t_{i+1}),$$

then

$$M_0 t_1 \ldots M_{i-1} t_i M_i t_{i+1} M_{i+1} \ldots t_n M_n \sim_N M_0 t_1 \ldots M_{i-1} t_{i+1} M_i' t_i M_{i+1} \ldots t_n M_n$$

where

$$\forall s \in S.\, M_i'(s) = M_{i-1}(s) + W(t_{i+1},s) - W(s,t_{i+1}).$$

(2) $\approx_N$ is the least equivalence relation containing $\sim_N$. □

We now show that these equivalence classes of occurrence sequences are again in one-to-one correspondence with the L-traces generated by a Petri net.

Lemma 4.7.2

Let $N = (S,T,W,M_{in})$ be a Petri net and let $\xi = M_0 t_1 M_1 t_2 \ldots t_n M_n$ and $\xi' = M_0' t_1' M_1' t_2' \ldots t_n' M_n'$ with $n \geq 0$ be occurrence sequences. Then

$$\xi \approx_N \xi' \Leftrightarrow t_1 \ldots t_n \cong_N t_1' \ldots t_n'.$$

Proof.

In order to prove the "only if"-part of the lemma it is sufficient to prove that $\xi \sim_N \xi'$ implies that $t_1 \ldots t_n \doteq_N t_1' \ldots t_n'$. So suppose that $\xi \sim_N \xi'$. Then there exists $1 \leq i \leq n-1$ such that $t_1' \ldots t_n' = t_1 \ldots t_{i-1} t_{i+1} t_i t_{i+2} \ldots t_n$ and $M_{i-1}(s) \geq W(s,t_i) + W(s,t_{i+1})$ for all $s \in S$. This implies that $t_1 \ldots t_{i-1}(t_i + t_{i+1}) \in MFS$ and hence also $(t_1 \ldots t_{i-1}, t_i + t_{i+1}) \in I_N$. Thus $t_1 \ldots t_n \doteq_N t_1' \ldots t_1'$.

Now in order to prove the "if"-part, it is sufficient to prove that if $t_1 \ldots t_n \doteq_N t_1' \ldots t_n'$, then $\xi \approx_N \xi'$. If $t_1 \ldots t_n = t_1' \ldots t_n'$ then this is trivial because then also $\xi = \xi'$, so assume that $t_1 \ldots t_n \neq t_1' \ldots t_n'$. Then there exists $1 \leq i \leq n-1$ such that $t_1' \ldots t_n' = t_1 \ldots t_{i-1} t_{i+1} t_i t_{i+2} \ldots t_n$ and $(t_1 \ldots t_{i-1}, \{t_i, t_{i+1}\}) \in I_N$. This implies that $M_{i-1}(s) \geq W(s,t_i) + W(s,t_{i+1})$ for all $s \in S$, and so $\xi \sim_N \xi'$. □

Theorem 4.7.3

Let $N = (S,T,W,M_{in})$ be a Petri net. Then there exists a bijection between $nl(N)$ and the set of equivalence classes of occurrence sequences associated with N.

Proof.

Follows immediately from Lemma 4.7.2 by observing that for every $\rho \in MFS$ there exists $\rho' \in FS$ such that $\rho \cong_N \rho'$. □

Thus whereas for 1-safe Petri nets there is a one-to-one correspondence between its M-traces and its processes, for general Petri nets there is a one-to-one correspondence between the equivalence classes of its processes (via the "swapping" operation) and its L-traces.

In [59] an algebraic characterization of the runs of a Petri net is given. This is done by performing a closure on the arrows of the graph representation of a Petri net with respect to an operation ; for sequential composition and an operation $\oplus$ for parallel composition. The runs are then represented as equivalence classes of the resulting arrows where the equivalence relation is defined in such a way that it captures the intended interpretation of the operations ; and $\oplus$. In [18] it has been shown that these equivalence classes of arrows coincide with the equivalence classes of processes from [9], and hence also with our L-traces.

One of the main differences between the approaches from [9] and [18] and our approach is that in the former approaches there is still a notion of a distributed state, while our trace semantics abstracts from this distribution. As will be shown in the next chapter, such an abstraction to a global state makes it possible to obtain also a branching time semantics for Petri nets in terms of event structures.

Chapter 5

An Event Structure Semantics for Petri Nets

In the previous chapter a trace semantics for Petri nets has been defined by associating an L-trace language with every Petri net. This trace semantics is based on representations of the finite runs of a Petri net, in which conflicts are resolved. Even though the branching aspects of a Petri net can be recovered from its L-traces by ordering them under a prefix ordering (resulting in the L-trace behaviour of the Petri net), no distinction is made between "different" occurrences of the same transition.

The aim of this chapter is to give a branching time semantics for Petri nets by associating a single object with each Petri net in which the relationship between different occurrences of transitions, called *events* is explicitly represented. Such objects are generically called *event structures*.

In [66] *prime event structures* have been introduced and it has been shown how they can be used to represent the behaviour of 1-safe Petri nets. In [66] also a map from prime event structures to 1-safe Petri nets has been defined. Later it was then shown by Winskel [96] that both maps can be extended to functors, forming a co-reflection.

In this chapter we propose a generalization of the prime event structure semantics for 1-safe Petri nets. To this aim we define a new class of event structures, called *local event structures*, and show that a subclass of local event structures with a certain unique occurrence property can be used to represent the behaviour of Petri nets. For associating a local event structure with each Petri net, it is however necessary to filter out auto-concurrency from the behaviour of Petri nets. In this sense the proposed event structure semantics is a restricted one.

In Section 5.1 we recall the classical construction from 1-safe Petri nets to prime event structures. Then in Section 5.2 we introduce local event structures and structure-preserving morphisms between them. In Section 5.3 a new equivalence relation over prime intervals is defined and this equivalence relation is used to define the unique occurrence property. Then in Section 5.4 a map from Petri nets to local event structures with the unique occurrence property is defined. In defining this map we use the set of step firing sequences rather than the set of multiset firing sequences of a Petri net. It is in this sense that we filter out auto-concurrency. In Section 5.5 we show that the class of local event structures yielded by Petri nets is exactly the class of local event

structures with the unique occurrence property. In proving this we use the regional construction defined in Section 3.2. Then in Section 5.6 we prove that when restricted to 1-safe Petri nets, our event structure semantics for Petri nets agrees with the prime event structure semantics for 1-safe Petri nets as given in Section 5.1.

In Section 5.7 we argue with the help of an example that the co-reflection result of Winskel will not go through in the present setting. The reason is that, due to auto-concurrency, the category of Petri nets is too rich in terms of objects and arrows. We show that the desired co-reflection *does* go through if we restrict our attention to either co-safe Petri nets or co-injective PN-morphisms. In Section 5.8 we discuss a generalization of local event structures, called *multiset event structures*, which might lead to a satisfactory event structure semantics for the category of *all* Petri nets. We show that there exists an adjunction between the category of multiset event structures with the unique occurrence property and the category of Petri nets. This adjunction is however not a co-reflection. Finally, in Section 5.9 we have some concluding remarks and mention some related work.

This chapter is based on [43], of which [42] is an extended abstract.

5.1 Prime Event Structures and 1-Safe Petri Nets

In this section a brief account is given of the prime event structure semantics for 1-safe Petri nets. We follow here the approach from [98] which differs from the original set-up in [66] by its explicit use of Mazurkiewicz' trace theory. However, rather than defining a map from arbitrary M-trace languages to prime event structures as in [98], we only apply their construction to the M-trace languages associated with 1-safe Petri nets.

First we introduce prime event structures using the definition from [97, 98].

Definition 5.1.1

A *prime event structure* is a triple $(E, \leq, \#)$ where E is a set of *events*, $\leq \subseteq E \times E$ is a partial order, the *causal dependency relation*, and $\# \subseteq E \times E$ is a symmetric, irreflexive relation, the *conflict relation*, satisfying

(P1) $e_0 \# e_1 \leq e_2 \Rightarrow e_0 \# e_2$

(P2) $\forall e \in E.\ \downarrow e$ is finite. □

Thus, by (P1), conflicts between events are inherited via the causal dependency relation. The condition (P2) states that each event has a finite cause. Note that by (P1) two causally related events are never in conflict.

With each prime event structure $P = (E, \leq, \#)$, a binary *concurrency relation* $co_P \subseteq E \times E$ can be associated which relates those events that are neither in conflict, nor causally related. Thus

$$e\ co_P\ e' \Leftrightarrow \neg(e \leq e' \text{ or } e' \leq e \text{ or } e \# e').$$

It is interesting to observe that co_P is a symmetric and irreflexive binary relation, and hence an M-independence relation over E.

A *configuration* of a prime event structure $P = (E, \leq, \#)$ is a subset of events which contains for each of its events the cause of that event, and which does not contain conflicting events.

In order to give a formal definition, let $c \subseteq E$. We say that c is *downward-closed* iff

$$\forall e, e' \in E.\, ((e \in c \text{ and } e' \leq e) \Rightarrow e' \in c).$$

We say that c is *#-free* iff

$$(c \times c) \cap \# = \emptyset.$$

Then c is a configuration iff c is downward-closed and #-free. We use C_P to denote the set of configurations of P and FC_P to denote the set $C_P \cap P_F(E)$ of all *finite* configurations of P.

As immediate consequences of the above, we have that $\downarrow e \in FC_P$ for each $e \in E$, and whenever $c, c' \in C_P$ are such that $(c \times c') \cap \# = \emptyset$, then also $c \cup c' \in C_P$.

Given a 1-safe Petri net $N = (S, T, W, M_{in})$ with independence relation $Ind_N \subseteq T \times T$, we now outline a method to define a prime event structure $sp(N)$ which captures the branching and concurrent aspects of the behaviour of N.

The events of $sp(N)$ correspond to occurrences of transitions in N. Different occurrences of one transition should give rise to different events. Whether or not two occurrences of one transition should be considered to be different, depends on the histories (i.e. runs of the net) leading to those occurrences. Two occurrences of one transition which have the same history represent the same event. Also two occurrences of a transition of which the histories only differ in the order of occurrence of concurrent transitions should be considered the same.

Transition occurrences are represented in terms of firing sequences. Then an equivalence relation is defined over these occurrences on the basis of Ind_N, the independence relation associated with a 1-safe Petri net N as given in Definition 3.4.8, and the induced M-equivalence relation $\simeq_N$.

Let

$$sPI_N = \{\rho t \mid \rho t \in FS\}$$

be the set of *sequential prime intervals* of N. So sPI_N is the set of non-empty firing sequences of N. A sequential prime interval ρt represents the occurrence of transition t after the firing sequence ρ.

The equivalence relation $\sim_N \subseteq sPI_N \times sPI_N$ is defined as the least equivalence relation which satisfies the following conditions (S1) and (S2).

(S1) $(\rho t' t \in FS \text{ and } (t, t') \in Ind_N) \Rightarrow \rho t \sim_N \rho t' t$.

(S2) $(\rho t, \rho' t \in sPI_N \text{ and } \rho \simeq_N \rho') \Rightarrow \rho t \sim_N \rho' t$.

Thus (S1) identifies sequential prime intervals which are connected by concurrent steps as given by the independence relation Ind_N. The condition (S2) identifies sequential prime intervals which have equivalent "pasts" under the M-equivalence relation $\sim_N$. The conditions (S1) and (S2) together ensure that equivalence of sequential prime intervals under $\sim_N$ is completely determined by the "diamonds" in the M-trace behaviour of N.

If N is clear from the context then we may omit the subscript $_N$ in $\sim_N$. In what follows we let for $\rho t \in sPI_N$, $\langle \rho t \rangle_{\sim_N}$ denote the equivalence class (under $\sim_N$) containing ρt, that is

$$\langle \rho t \rangle_{\sim_N} = \{\rho' t' \in sPI_N \mid \rho' t' \sim_N \rho t\}.$$

The equivalence classes $\langle \rho t \rangle_{\sim_N}$ will be the events of $sp(N)$.

In order to define the causal dependency and conflict relation of $sp(N)$ we need the following function $ev_N : FS \to P_F(sPI_N/\sim_N)$ which associates with each firing sequence of N its set of events:

- $ev_N(\Lambda) = \emptyset$
- $ev_N(\rho t) = ev_N(\rho) \cup \{\langle \rho t \rangle_{\sim_N}\}$.

As the following lemma from [98] shows, all events of a firing sequence ρ have a unique representative in ρ. In this sense $ev_N(\rho)$ is a faithful representation of the events in ρ.

Lemma 5.1.2

Let N be a 1-safe Petri net and let $\rho t \in sPI_N$. Then $\langle \rho t \rangle_{\sim_N} \notin ev_N(\rho)$. □

The next lemma, also from [98], shows that the branching behaviour of a 1-safe Petri net N as given by the M-trace ordering of its M-traces is also captured by the function ev_N: the M-traces of two firing sequences are ordered iff their associated sets of events are ordered (under inclusion).

Lemma 5.1.3

Let N be a 1-safe Petri net and let $\rho, \rho' \in FS$. Then

$$[\rho]_{Ind_N} \preceq [\rho']_{Ind_N} \Leftrightarrow ev_N(\rho) \subseteq ev_N(\rho').$$

□

Thus in particular $\rho \simeq \rho'$ iff $ev_N(\rho) = ev_N(\rho')$ for two firing sequences ρ and ρ' of a 1-safe Petri net N.

Given a 1-safe Petri net N, a causal dependency relation and a conflict relation over the transitions are defined in the following way.

An event e_2 causally depends on an event e_1 if e_2 cannot occur without e_1 having occurred first. That is, no firing sequence of N has e_2 as an event without having e_1 as one of its events. The events e_1 and e_2 are in conflict if they cannot occur "together". That is, no firing sequence of N has both e_1 and e_2 in its set of events.

Thus formally $sp(N)$ is defined as follows.

Definition 5.1.4

Let N be a 1-safe Petri net. Then $sp(N) = (E, \leq, \#)$ is the *prime event structure associated with* N where

- $E = \{\langle \rho t \rangle_{\sim_N} \mid \rho t \in sPI_N\}$

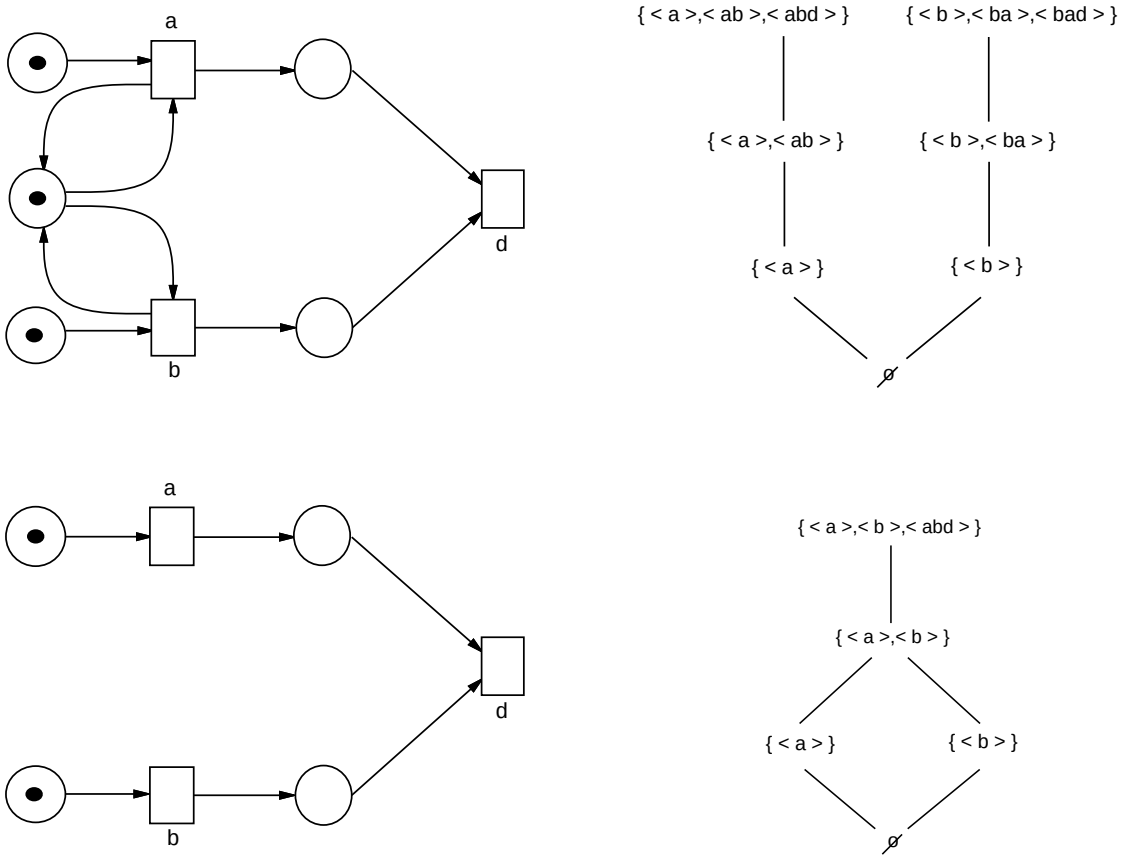


Figure 5.1: The 1-safe Petri nets N_3 and N_4 with their associated prime event structures

- $e \leq e' \Leftrightarrow \forall \rho \in FS_N.(e' \in ev_N(\rho) \Rightarrow e \in ev_N(\rho))$
- $e \# e' \Leftrightarrow \forall \rho \in FS_N.(e \in ev_N(\rho) \Rightarrow e' \notin ev_N(\rho))$. □

Example 5.1.5

In Figure 5.1 the 1-safe Petri nets N_3 and N_4 from Figure 2.3 and Figure 2.4 are depicted together with their associated prime event structures. The prime event structures are depicted through their finite configurations, ordered under inclusion. Both Petri nets have the same set of sequential prime intervals: $sPI_{N_3} = sPI_{N_4} = \{a, b, ab, ba, abd, bad\}$. For N_3 these sequential prime intervals all generate different events. For N_4 on the other hand, $(a, b) \in Ind_{N_4}$ implies that $a \sim_{N_4} ba$ and $b \sim_{N_4} ab$ by (S1). Since $(a, b) \in Ind_{N_4}$ also implies that $ab \simeq_{N_4} ba$, we also have that $abd \sim_{N_4} bad$ by (S2). □

For the Petri nets N_3 and N_4 considered in the previous example the poset as given by the M-trace behaviour (see Figure 4.1) is isomorphic to the poset of the (finite) configurations of the associated prime event structure (ordered under inclusion). It turns out that this holds for every 1-safe Petri net. This follows immediately from

Lemma 5.1.3 and the following lemma [98] which characterizes the finite configurations of the prime event structure associated with a 1-safe Petri net.

Lemma 5.1.6
Let N be a 1-safe Petri net with $sp(N) = (E, \leq, \#)$. Then

$$FC_{sp(N)} = \{ev_N(\rho) \mid \rho \in FS\}.$$

□

Finally we note that also concurrency in a 1-safe Petri net is easily derivable from its associated prime event structure. This follows from the next lemma (see [98]) stating that two events are in the concurrency relation of the prime event structure iff the transitions generating these events are concurrent at some reachable marking.

Lemma 5.1.7
Let N be a 1-safe Petri net with $sp(N) = (E, \leq, \#)$ and let $\langle \rho t \rangle_{\sim_N}, \langle \rho' t' \rangle_{\sim_N} \in E$. Then

$$\langle \rho t \rangle_{\sim_N} co_{sp(N)} \langle \rho' t' \rangle_{\sim_N} \Leftrightarrow$$
$$\exists \rho'' t, \rho'' t' \in sPI_N. (\rho'' t \sim_N \rho t \text{ and } \rho'' t' \sim_N \rho' t' \text{ and } (t, t') \in Ind_N).$$

□

5.2 Local Event Structures

The approach described in the previous section for 1-safe Petri nets does not work for general Petri nets. As explained in Chapter 2, for general Petri nets conflict and concurrency are no global relations. This leads us to consider *local event structures* where concurrency is specified locally, i.e. per configuration. This is similar to the context-dependent concurrency underlying the definition of local trace languages in Section 4.2. The identification of the events associated with a Petri net leads however to some new problems which will be discussed in Section 5.3.

Whereas for prime event structures the set of (finite) configurations is a derived notion, a local event structure is defined directly as a family of configurations. This is similar to the specification of Winskel's general event structures through families of configurations (see [96] and Chapter 6 of this thesis). In addition to this, such a family of configurations of a local event structure is also equipped with an enabling relation which specifies *locally*, for each configuration, the possible concurrency of (sets of) events at that configuration. This enabling relation satisfies some simple axioms.

Definition 5.2.1
A *local event structure* is a triple $ES = (E, C, \vdash)$ where E is a set of events, $C \subseteq P_F(E)$ is a non-empty set of *(finite) configurations* (ranged over by c,c', etc.), and $\vdash \subseteq C \times P_F(E)$ is an *enabling relation* satisfying:

(E0) $\emptyset \neq c \Rightarrow \exists e \in c. c - e \vdash e$.

(E1) $c \vdash \emptyset$.

(E2) $c \vdash u \Rightarrow (c \cap u = \emptyset \text{ and } \forall v \subseteq u.\, (c \vdash v \text{ and } c \cup v \vdash u - v))$. □

Note that (E0) implies that if $\emptyset \neq c \in C$, then there exists $e \in c$ such that $c - e \in C$. Hence $\emptyset \in C$, because C is non-empty. The axiom (E2) states that each event can occur at most once, and that steps can be split arbitrarily into substeps. The axiom (E2) also implies that if $c \vdash u$ then $c \cup v \in C$ for all $v \subseteq u$.

Note that the axiom (E1) could have been replaced by the condition that the enabling relation $\vdash$ is non-empty.

From now on we refer to local event structures as L-event structures.

Example 5.2.2

In Figure 5.2 three L-event structures ES_1, ES_2, and ES_3 are depicted. In depicting an L-event structure $(E, C, \vdash)$ we use the following convention. If $c \vdash u$ then we draw a line between c and $c \cup u$ in case $|u| = 1$ and we draw a dotted line between c and $c \cup u$ in case $|u| \geq 2$. □

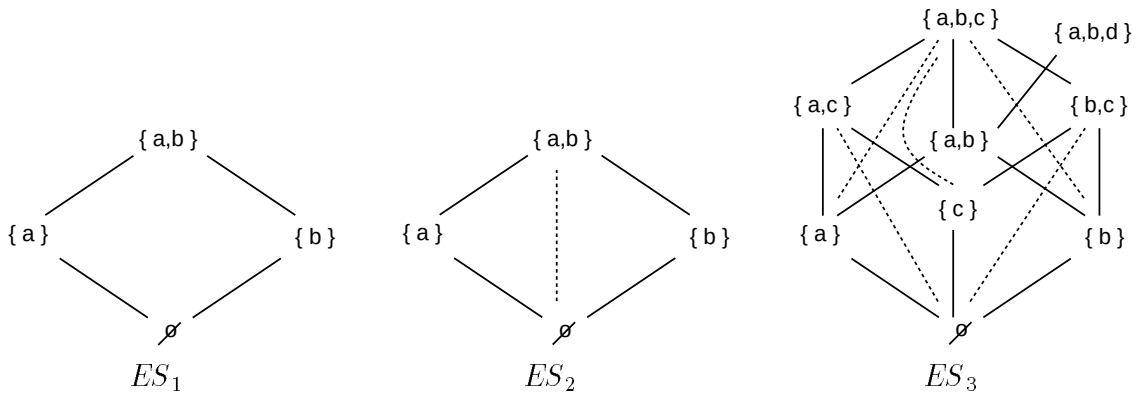


Figure 5.2: Three L-event structures

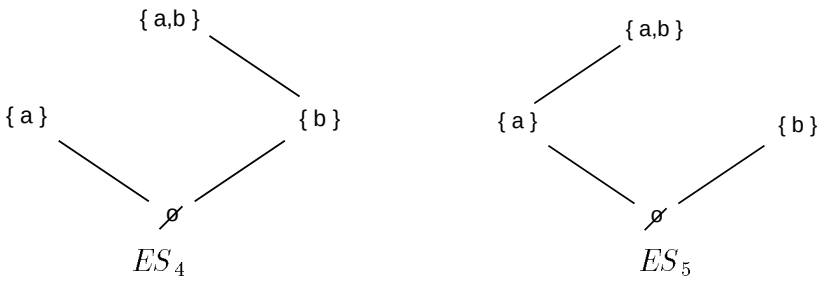


Figure 5.3: L-event structures with the same configurations

We would now like to establish some preliminary properties of L-event structures. Before doing so, we wish to emphasize that the inclusion relation between configurations in the present set-up does not carry much information. Consider the L-event structures ES_1 and ES_2 shown in Figure 5.2) and the L-event structures ES_4 and ES_5 depicted in Figure 5.3. Clearly the sets of configurations of these four L-event structures are identical. They all have however a different enabling relation.

A partial ordering relation describing reachability between the configurations of an L-event structure would carry more useful information. In order to define this relation, first a multiset transition diagram is associated with each L-event structure. The multiset transition relation is such that if a step is enabled at a configuration, then there is a transition from this configuration to the configuration obtained by adding the step to the original configuration.

Let $ES = (E, C, \vdash)$ be an L-event structure. Then $(C, E, \longrightarrow_{ES})$ is the *multiset transition diagram associated with ES* where $\longrightarrow_{ES}$ is given by:

$$c \xrightarrow{u}_{ES} c' \Leftrightarrow (c \vdash u \text{ and } c' = c \cup u).$$

Note that by our drawing conventions for L-event structures, we actually depict the multiset transition diagram associated with an L-event structure. Also note that there are no enablings of multisets of events, so that in fact $\longrightarrow_{ES} \subseteq C \times P_F(E) \times C$.

The *reachability relation* $\sqsubseteq_{ES} \subseteq C \times C$ is now defined by:

$$c \sqsubseteq_{ES} c' \Leftrightarrow \exists \rho \in (P_F(E))^+ . c \xrightarrow{\rho}_{ES} c'.$$

It is easy to see that the reachability relation $\sqsubseteq_{ES}$ is a partial ordering relation. In what follows we omit the subscript $_{ES}$ in $\sqsubseteq_{ES}$ if ES is clear from the context.

Lemma 5.2.3

Let $(E, C, \vdash)$ be an L-event structure and let $c \in C$ and $e_1, e_2 \in c$ be such that $e_1 \neq e_2$. Then

(1) $\exists c' \in C . c' \sqsubseteq c$ and $((e_1 \in c' \text{ and } c' \vdash e_2) \text{ or } (e_2 \in c' \text{ and } c' \vdash e_1))$

(2) $\exists c' \in C . c' \sqsubseteq c$ and $(e_1 \in c' \Rightarrow e_2 \notin c')$.

Proof.

In order to prove (1), we proceed by induction on $k = |c|$. If $k = 2$ then $c = \{e_1, e_2\}$ and by (E0), $c - e_1 \vdash e_1$ or $c - e_2 \vdash e_2$. In either case the required result follows.

If $k > 2$ then, again by (E0), there exists $e \in c$ such that $c - e \vdash e$. If $e = e_1$ or $e = e_2$ then let $c' = c - e$. Otherwise the required $c' \in C$ exists by the induction hypothesis applied to $c - e$.

(2) follows immediately from (1) and (E2). □

By Lemma 5.2.3(2) L-event structures satisfy a coincidence freeness property, similar to Winskel's general event structures [96].

By (E0), each configuration c of an L-event structure $ES = (E, C, \vdash)$ is reachable (in terms of $\sqsubseteq_{ES}$) from $\emptyset$. There may however be various routes from $\emptyset$ to c. Each such route is given by a sequence of transitions from $\longrightarrow_{ES}$. This motivates the introduction of the set SFS_{ES} of *step firing sequences* of ES and a function cf associating with each step firing sequence the configuration it leads to.

Definition 5.2.4
Let $ES = (E, C, \vdash)$ be an L-event structure. The set $SFS_{ES} \subseteq (P_F(E))^+$ of *step firing sequences* of ES and the function $cf_{ES} : SFS_{ES} \to P_F(E)$ are given inductively by:

- $\emptyset \in SFS_{ES}$ and $cf_{ES}(\emptyset) = \emptyset$
- $(\rho \in SFS_{ES}$ and $cf_{ES}(\rho) \vdash u) \Rightarrow (\rho u \in SFS_{ES}$ and $cf_{ES}(\rho u) = cf_{ES}(\rho) \cup u)$. □

If the L-event structure ES is clear from the context, then we may omit the subscript $_{ES}$ in SFS_{ES} and cf_{ES}.

In this way we obtain the following characterization of the configurations of an L-event structure.

Lemma 5.2.5
Let $(E, C, \vdash)$ be an L-event structure. Then

(1) $\forall \rho \in SFS. (cf(\rho) \in C$ and $cf(\rho) = alph(\rho))$

(2) $C = \{alph(\rho) \mid \rho \in SFS\}$

(3) $\forall \rho, \rho' \in SFS. (alph(\rho) = alph(\rho') \Rightarrow (\rho u \in SFS \Leftrightarrow \rho' u \in SFS))$

(4) $\forall \rho \in SFS. \forall e \in E. num_e(\rho) \leq 1.$

Proof.

(1) Let $\rho \in SFS$. The proof is by induction on $k = |\rho|$. If $k = 0$ then $\rho = \emptyset$ and hence $cf(\rho) = \emptyset \in C$ and $cf(\rho) = \emptyset = alph(\rho)$. Now assume that $k > 0$. Then there exist $\rho' \in SFS$ and $\emptyset \neq u \in P_F(E)$ such that $cf(\rho') \vdash u$ and $\rho = \rho' u$. Hence $cf(\rho) = cf(\rho') \cup u \in C$ by (E2) and $cf(\rho) = alph(\rho)$ by the induction hypothesis applied to ρ'.

(2) If $\rho \in SFS$ then $alph(\rho) = cf(\rho) \in C$ by (1). Now let $c \in C$. We proceed by induction on $k = |c|$. If $k = 0$ then $c = \emptyset$ and hence $\rho = \emptyset \in SFS$ is such that $alph(\rho) = c$. Now assume that $k > 0$. Then by (E0) there exists $e \in c$ such that $c - e \vdash e$. By the induction hypothesis applied to $c - e$ there exists $\rho' \in SFS$ such that $alph(\rho') = cf(\rho') = c - e$. Then $\rho' e \in SFS$ by the definition of SFS and $alph(\rho' e) = c$.

(3) Let $\rho, \rho' \in SFS$ be such that $alph(\rho) = alph(\rho')$. If $u = \emptyset$ then $\rho u, \rho' u \in SFS$ by (E1). If $u \neq \emptyset$ then $cf(\rho) = cf(\rho')$ by (1) and hence $\rho u \in SFS$ iff $cf(\rho) \vdash u$ iff $\rho' u \in SFS$.

(4) Let $\rho \in SFS$. The proof is by induction on $k = |\rho|$. If $k = 0$ then the claim clearly holds. Now assume that $k > 0$. Then there exist $\rho' \in SFS$ and $\emptyset \neq u \in P_F(E)$ such that $\rho = \rho' u$ and $cf(\rho') \vdash u$. Then $num_e(\rho') \leq 1$ for all $e \in E$ by the induction hypothesis applied to ρ'. Because $cf(\rho') \cap u = \emptyset$ by (E2) and $alph(\rho') = cf(\rho')$ by (1) we can now conclude that also $num_e(\rho) \leq 1$ for all $e \in E$. □

Consequently, for a given configuration c of an L-event structure ES, all step firing sequences leading to it have the same set of events occurring in them, namely c. Conversely, whenever two step firing sequences have the same set of events occurring in them, they lead to the same configuration.

As the last point of this section, we introduce structure-preserving morphisms between L-event structures.

Definition 5.2.6

Let $ES_i = (E_i, C_i, \vdash_i)$, $i = 1, 2$, be a pair of L-event structures. An *LES-morphism* from ES_1 to ES_2 is a partial function $f : E_1 \to E_2$ such that

$$\forall c \in C_1. \forall u \in P_F(E_1). (c \vdash_1 u \Rightarrow f(c) \vdash_2 f(u)).$$

□

Let, for an arbitrary L-event structure ES, id_{ES} denote the identity function on its events. Then an LES-morphism f from $ES_1 = (E_1, C_1, \vdash_1)$ to $ES_2 = (E_2, C_2, \vdash_2)$ is an *LES-isomorphism* iff there exists an LES-morphism g from ES_2 to ES_1 such that $g \circ f = id_{ES_1}$ and $f \circ g = id_{ES_2}$. It is easy to see that f is an LES-isomorphism from ES_1 to ES_2 iff

(1) f is a bijection

(2) $c \vdash_1 u \Leftrightarrow f(c) \vdash_2 f(u)$.

If ES_1 and ES_2 are LES-isomorphic then this is denoted by $ES_1 \equiv ES_2$.

We conclude with some properties of LES-morphisms which will be useful in later sections.

The first property shows that LES-morphisms are injective on concurrent steps.

Lemma 5.2.7

Let f be an LES-morphism from $(E_1, C_1, \vdash_1)$ to $(E_2, C_2, \vdash_2)$ and let $c \in C_1$ and $e_1, e_2 \in c$ be such that $e_1 \neq e_2$ and both $f(e_1)$ and $f(e_2)$ are defined. Then $f(e_1) \neq f(e_2)$.

Proof.

By Lemma 5.2.3(1) we may assume without loss of generality that there exists $c' \sqsubseteq c$ such that $e_1 \in c'$ and $c' \vdash_1 e_2$. By the definition of an LES-morphism we then have $f(c') \vdash_2 f(e_2)$ and so $f(e_2) \notin f(c')$ by (E2), and $f(e_1) \in f(c')$. □

Secondly, LES-morphisms are behaviour-preserving with respect to step firing sequences.

Lemma 5.2.8

Let f be an LES-morphism from $ES_1 = (E_1, C_1, \vdash_1)$ to $ES_2 = (E_2, C_2, \vdash_2)$. Then $f(SFS_{ES_1}) \subseteq SFS_{ES_2}$.

Proof.

Let $\rho \in SFS_{ES_1}$. We prove by induction on $|\rho|$ that $f(\rho) \in SFS_{ES_2}$. If $\rho = \emptyset$ then this is clear, so assume that there exist $\rho' \in SFS_{ES_1}$ and $\emptyset \neq u \in P_F(E_1)$ such that $\rho = \rho' u$. Then $alph(\rho') \vdash_1 u$. Hence $f(alph(\rho')) \vdash_2 f(u)$ because f is an LES-morphism. Since $f(\rho') \in SFS_{ES_2}$ by the induction hypothesis and $f(alph(\rho')) = alph(f(\rho'))$ this implies that $f(\rho')f(u) = f(\rho) \in SFS_{ES_2}$. □

5.3 Equivalence of Prime Intervals

In this section a new equivalence relation over prime intervals is defined. This equivalence relation will be used in Section 5.4 for defining a map from Petri nets to L-event structures. In this section the equivalence relation is used for defining the *unique occurrence property* of L-event structures.

For a 1-safe Petri net the events in its associated prime event structure can be extracted from the M-trace behaviour as equivalence classes of sequential prime intervals. The equivalence of sequential prime intervals is then determined by the "diamonds" formed by concurrent steps in the M-trace behaviour. This construction ensures that the prime event structure associated with a 1-safe Petri net is deterministic in the sense that at each configuration different enabled events correspond to different transitions of the Petri net.

For general Petri nets, a similar approach with L-traces instead of M-traces does not work. Consider, e.g., the L-trace behaviour of the Petri net N_6 as depicted in Figure 4.5. Then all prime intervals corresponding to a are connected by diamonds, and also all prime intervals corresponding to b are connected by diamonds. Now consider the Petri net N_{10} depicted in Figure 5.4 together with its L-trace behaviour. This Petri net has a transition d added so that d can only occur if both a and b have occurred, but c has not yet occurred. The two prime intervals corresponding to d are not connected by diamonds so they would correspond to different events. On the other hand, the history of both occurrences is the same because the prime intervals corresponding to a are equivalent and the prime intervals corresponding to b are equivalent. This then leads to non-determinism in the sense that at the configuration corresponding to this history two different events are enabled, while both correspond to the same transition d of the Petri net.

The above considerations lead us to consider in this section a new equivalence relation over prime intervals. The equivalence relation we will use to identify occurrences of a transition is, like condition (S1) in Section 5.1, still based on the diamond property of concurrency. But instead of lifting simply the M-trace equivalence of (S2), used to

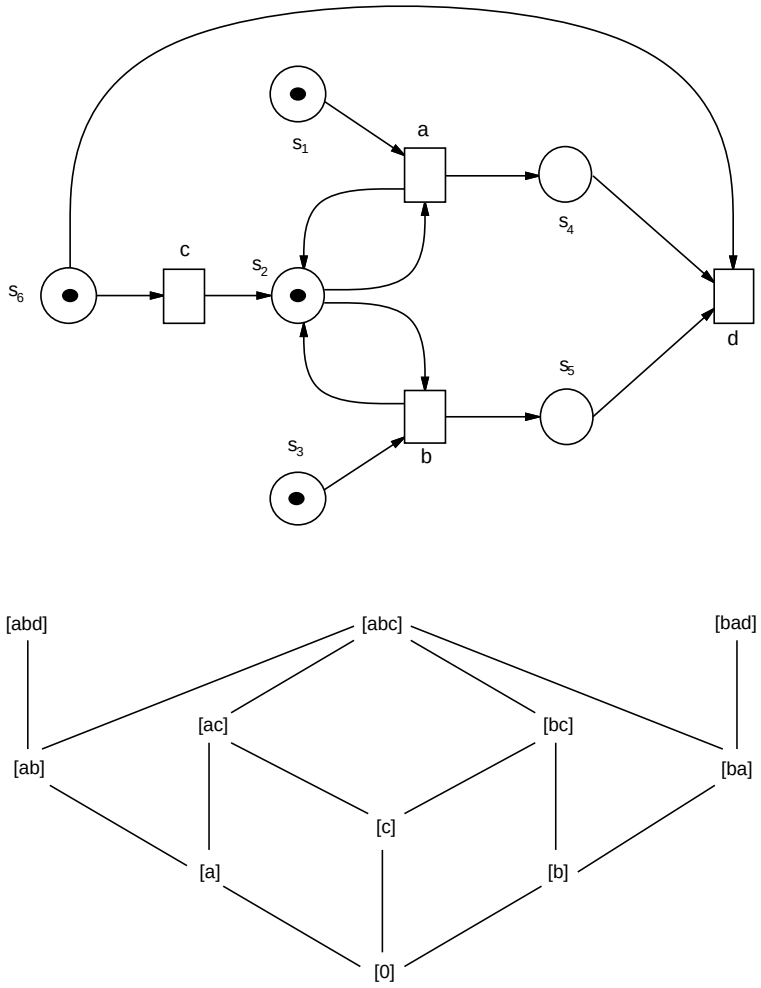


Figure 5.4: The Petri net N_{10} with its L-trace behaviour

identify transition occurrences with equivalent histories, to the level of L-trace equivalence, we have to follow a more complicated approach. The idea is still to identify transition occurrences with equivalent histories. But now it is taken into account that this equivalence may not be a direct consequence of concurrency of some events in these histories, but caused by possible concurrency in another context. In Section 5.4 this equivalence relation is then used for defining an event structure semantics for Petri nets.

The equivalence relation is here however not defined in terms of the (multiset) firing sequences of Petri nets, but rather in the abstract setting of step sequences, because the same equivalence relation is also used in this section for defining the crucial unique occurrence property of L-event structures. Whereas in general an event in an L-event structure may have different (non-equivalent) occurrences, the unique occurrence property states that there is a bijection between the events in an L-event structure and their occurrences. In other words, each event has a unique occurrence.

Instead of using sequential prime intervals as in Section 5.1, we use here prime intervals with respect to step sequences.

In order to define the equivalence relation, let X be an alphabet and let $L \subseteq (P_F(X))^+$ be a set of step sequences satisfying the following two conditions.

(L1) $\rho u \in L \Rightarrow \rho \in L$.

(L2) $\rho u \in L \Rightarrow \forall v \subseteq u.\, \rho v(u-v) \in L$.

The set of *prime intervals* of L, denoted by PI_L, is given by:

$$PI_L = \{\rho a \mid \rho a \in L\}.$$

Let $int_L : L \to P_F(PI)$ be the function which maps each step sequence to the set of prime intervals in that sequence. Thus int_L is given inductively by:

- $int_L(\emptyset) = \emptyset$
- $int_L(\rho u) = int_L(\rho) \cup \{\rho a \mid a \in u\}$.

Note that int_L is well-defined, because if $\rho u \in L$, then also $\rho \in L$ by (L1) and $\rho a \in L$ for all $a \in u$ by (L2) and (L1). If L is clear from the context, then we may omit the subscript $_L$ in PI_L and int_L.

Our desired equivalence relation over prime intervals should be *L-consistent* in the sense that it identifies prime intervals which are connected by concurrent steps and that it identifies prime intervals with the same history.

Given an arbitrary equivalence relation $R \subseteq PI \times PI$, let for $\rho a \in PI$, $\langle \rho a \rangle_R$ be the equivalence class (under R) containing ρa, that is

$$\langle \rho a \rangle_R = \{\rho' a' \in PI \mid \rho' a' \, R \, \rho a\}.$$

Furthermore, let $past_R : L \to P_F(PI/R)$, the function which maps each step sequence to the set of equivalence classes of prime intervals in that sequence, be given by:

$$past_R(\rho) = \{\langle \rho' a \rangle_R \mid \rho' a \in int(\rho)\}.$$

Definition 5.3.1

Let X be an alphabet and let $L \subseteq (P_F(X))^+$ be a set of step sequences satisfying (L1) and (L2). Then an equivalence relation $R \subseteq PI \times PI$ is *L-consistent* if it satisfies the following conditions.

(C1) $(\rho u \in L \text{ and } a \in u) \Rightarrow \rho a \, R \, \rho(u-a)a$.

(C2) $\rho a, \rho' a \in PI \Rightarrow (past_R(\rho) = past_R(\rho') \Rightarrow \rho a \, R \, \rho' a)$. □

Note that the condition (C1) in the above definition is well-defined, because whenever $\rho u \in L$ and $a \in u$, then by (L2) $\rho a(u-a), \rho(u-a)a \in L$ and hence by (L1) also $\rho a \in L$.

The condition (C2) demands that prime intervals $\rho a, \rho' a$ which have R-equivalent pasts in the sense that the same R-equivalent prime intervals occur in ρ and ρ', should in turn be R-equivalent.

In general there may be infinitely many equivalence relations which are L-consistent.

Lemma 5.3.2

Let X be an alphabet, let $L \subseteq (P_F(X))^+$ be a set of step sequences satisfying (L1) and (L2), and let $K = \{R \subseteq PI \times PI \mid R \text{ is an } L\text{-consistent equivalence relation }\}$. Then $K \neq \emptyset$ and $\bigcap K \in K$.

Proof.

Since $PI \times PI$ is clearly an equivalence relation which is L-consistent, we have that $K \neq \emptyset$.

Now let $\hat{R} = \bigcap K$. Then it is clear that $\hat{R}$ is an equivalence relation. Suppose $\rho u \in L$ and $a \in u$. Then $\rho a \, R \, \rho(u-a)a$ for all $R \in K$ because each $R \in K$ satisfies (C1). Hence also $\rho a \, \hat{R} \, \rho(u-a)a$.

In order to prove that $\hat{R}$ satisfies (C2), let $\rho a, \rho' a \in PI$ be such that $past_{\hat{R}}(\rho) = past_{\hat{R}}(\rho')$. It suffices to prove that $past_R(\rho) = past_R(\rho')$ for every $R \in K$. Because in that case $\rho a \, R \, \rho' a$ for every $R \in K$ and hence $\rho a \, \hat{R} \, \rho' a$.

So, let $R \in K$ and suppose $\langle \rho_1 a_1 \rangle_R \in past_R(\rho)$. Then there exists $\rho_2 a_2 \in int(\rho)$ such that $\langle \rho_1 a_1 \rangle_R = \langle \rho_2 a_2 \rangle_R$. We then also have that $\langle \rho_2 a_2 \rangle_{\hat{R}} \in past_{\hat{R}}(\rho) = past_{\hat{R}}(\rho')$. Then there exists $\rho_3 a_3 \in int(\rho')$ such that $\langle \rho_2 a_2 \rangle_{\hat{R}} = \langle \rho_3 a_3 \rangle_{\hat{R}}$. Hence also $\langle \rho_3 a_3 \rangle_R \in past_R(\rho')$. Moreover, $\langle \rho_2 a_2 \rangle_R = \langle \rho_3 a_3 \rangle_R$ because $\hat{R} \subseteq R$. This proves that $\langle \rho_1 a_1 \rangle_R \in past_R(\rho')$. Similarly it can be proved that $past_R(\rho') \subseteq past_R(\rho)$.

This proves that $past_R(\rho) = past_R(\rho')$ for all $R \in K$. □

Given an alphabet X and a set of step sequences $L \subseteq (P_F(X))^+$ satisfying (L1) and (L2), there exists by the above lemma a least equivalence relation contained in $PI \times PI$ which is L-consistent. This equivalence relation is denoted by $\hat{R}$ in the proof of Lemma 5.3.2.

Definition 5.3.3

Let X be an alphabet, let $L \subseteq (P_F(X))^+$ be a set of step sequences satisfying (L1) and (L2). Then $\approx_L \subseteq PI \times PI$ is the least equivalence relation which is L-consistent. □

In what follows we write $\langle \rho a \rangle_L$ and $past_L$ rather than $\langle \rho a \rangle_{\approx_L}$ and $past_{\approx_L}$ respectively. If $\approx_L$ is the only equivalence relation under consideration, then we may even omit the subscript $_L$.

Lemma 5.3.4

Let X be an alphabet, let $L \subseteq (P_F(X))^+$ be a set of step sequences satisfying (L1) and (L2), and let $\rho_1 a_1, \rho_2 a_2 \in PI$ be such that $\rho_1 a_1 \approx_L \rho_2 a_2$. Then

(1) $a_1 = a_2$ and $num_{a_1}(\rho_1) = num_{a_2}(\rho_2)$

(2) $\rho_1 a_1 \approx_{L'} \rho_2 a_2$ whenever $L' \subseteq (P_F(X))^+$ is such that L' satisfies (L1) and (L2) and $L \subseteq L'$.

Proof.

In order to prove (1), define the equivalence relation $R \subseteq PI \times PI$ by: $\rho a \, R \, \rho' a'$ iff $a = a'$ and $num_a(\rho) = num_{a'}(\rho')$. It is sufficient to prove that R is L-consistent. Then the required result follows from the fact that $\approx_L \subseteq R$.

Clearly, R satisfies (C1). Let $\rho a, \rho' a \in PI$ be such that $past_R(\rho) = past_R(\rho')$. We first want to argue that $num_a(\rho') \geq num_a(\rho)$. If $num_a(\rho) = 0$ then this is trivial, so assume that $num_a(\rho) > 0$. Then there exists $\rho_1 a \in int(\rho)$ such that $num_a(\rho_1) = num_a(\rho) - 1$. Then $\langle \rho_1 a \rangle_R \in past_R(\rho) = past_R(\rho')$. Hence there exists $\rho_2 a \in int(\rho')$ such that $\langle \rho_1 a \rangle_R = \langle \rho_2 a \rangle_R$ which implies that $num_a(\rho_1) = num_a(\rho_2)$. We now have $num_a(\rho') \geq num_a(\rho_2) + 1 = num_a(\rho_1) + 1 = num_a(\rho)$. Similarly we can prove that $num_a(\rho') \leq num_a(\rho)$ and thus $num_a(\rho) = num_a(\rho')$. Consequently $\rho a \, R \, \rho' a$ which implies that R satisfies (C2).

Now in order to prove (2), let $L' \subseteq (P_F(X))^+$ be such that $L \subseteq L'$ and L' satisfies (L1) and (L2).

Define the equivalence relation $R \subseteq PI_L \times PI_L$ by: $\rho a \, R \, \rho' a'$ iff $\rho a \approx_{L'} \rho' a'$. It is sufficient to prove that R is L-consistent because then $\approx_L \subseteq R$.

Clearly, R satisfies (C1). In order to prove (C2), let $\rho a, \rho' a \in PI_L$ be such that $past_R(\rho) = past_R(\rho')$. It is sufficient to show that $past_{L'}(\rho) = past_{L'}(\rho')$, because $\approx_{L'}$ satisfies (C2).

Let $\langle \rho_3 a_3 \rangle_{L'} \in past_{L'}(\rho)$. Then there exists $\rho_4 a_4 \in int_{L'}(\rho) = int_L(\rho)$ with $\langle \rho_3 a_3 \rangle_{L'} = \langle \rho_4 a_4 \rangle_{L'}$. Then also $\langle \rho_4 a_4 \rangle_R \in past_R(\rho) = past_R(\rho')$. Hence there exists $\rho_5 a_5 \in int_L(\rho') = int_{L'}(\rho')$ with $\langle \rho_4 a_4 \rangle_R = \langle \rho_5 a_5 \rangle_R$. Then $\rho_4 a_4 \approx_{L'} \rho_5 a_5$ by the definition of R. Moreover, $\langle \rho_5 a_5 \rangle_{L'} \in past_{L'}(\rho')$. This proves that $\langle \rho_3 a_3 \rangle_{L'} \in past_{L'}(\rho')$. Similarly it can be proved that $past_{L'}(\rho') \subseteq past_{L'}(\rho)$ and thus $past_{L'}(\rho) = past_{L'}(\rho')$. □

By Lemma 5.1.2, the events associated with a 1-safe Petri net can occur at most once in a firing sequence. For the equivalence classes of $\approx_L$ we have a similar result.

Lemma 5.3.5

Let X be an alphabet, let $L \subseteq (P_F(X))^+$ be a set of step sequences satisfying (L1) and (L2), and let $\rho a \in PI$. Then $\langle \rho a \rangle_L \notin past_L(\rho)$.

Proof.

Assume to the contrary that $\langle \rho a \rangle_L \in past_L(\rho)$. Then there exists $\rho' a' \in int(\rho)$ with $\rho a \approx_L \rho' a'$. Then by Lemma 5.3.4(1), $a = a'$ and $num_a(\rho) = num_{a'}(\rho')$. On the other hand, $\rho' a' \in int(\rho)$ implies that $num_{a'}(\rho') < num_{a'}(\rho)$, a contradiction. Hence we must have that $\langle \rho a \rangle_L \notin past_L(\rho)$. □

To conclude this section we now use the equivalence relation $\approx_L$ to define the unique occurrence property for L-event structures. This unique occurrence property states that for each L-event structure ES satisfying this property and for each event of this L-event structure, all occurrences of this event are essentially the same. By the following lemma, which follows immediately from the definition of the set of step firing sequences of an L-event structure, we may indeed use the equivalence relation $\approx_{SFS_{ES}}$.

Lemma 5.3.6

Let $ES = (E, C, \vdash)$ be an L-event structure. Then $SFS_{ES} \subseteq (P_F(E))^+$ satisfies the conditions (L1) and (L2). □

Hence given an L-event structure ES we have the equivalence relation $\approx_{SFS_{ES}}$. In what follows we write PI_{ES}, int_{ES}, $\approx_{ES}$, $\langle \rho e \rangle_{ES}$, and $past_{ES}$ rather than $PI_{SFS_{ES}}$, $int_{SFS_{ES}}$, $\approx_{SFS_{ES}}$, $\langle \rho e \rangle_{\approx_{ES}}$, and $past_{\approx_{ES}}$ respectively.

The unique occurrence property of local event structures is now defined in terms of the equivalence relation $\approx_{ES}$.

Definition 5.3.7

An L-event structure $ES = (E, C, \vdash)$ has the *unique occurrence property* if

(U1) $\forall e \in E. \exists \rho e \in PI_{ES}$

(U2) $\forall \rho_1 e, \rho_2 e \in PI_{ES}. \rho_1 e \approx_{ES} \rho_2 e.$ □

From now on L-event structures satisfying the unique occurrence property will be referred to as UL-event structures.

The condition (U1) guarantees that each event of an UL-event structure ES has at least one occurrence, while condition (U2) states that all occurrences of an event are the same (under $\approx_{ES}$). Thus for a UL-event structure ES there exists a bijective correspondence between its events and the equivalence classes of its prime intervals under $\approx_{ES}$.

As the following lemma shows, the unique occurrence property is preserved under LES-isomorphisms.

Lemma 5.3.8

Let ES_i, $i = 1, 2$, be a pair of L-event structures such that $ES_1 \equiv ES_2$. Then ES_1 has the unique occurrence property iff ES_2 has the unique occurrence property.

Proof.

Follows easily from Lemma 5.2.8. □

Example 5.3.9

From the L-event structures from Example 5.2.2, ES_1 is not an UL-event structure. Both ES_2 and ES_3 are UL-event structures. In ES_3, $bc \approx_{ES_3} c$ and $cb \approx_{ES_3} b$ by (C1), and hence $past_{ES_3}(bc) = past_{ES_3}(cb)$. This implies that $bca \approx_{ES_3} cba$ by (C2). Then $a \approx_{ES_3} ca \approx_{ES_3} cba \approx_{ES_3} bca \approx_{ES_3} ba$ by (C1). Similarly, $b \approx_{ES_3} ab$, and hence $past_{ES_3}(ab) = past_{ES_3}(ba)$. Now $abd \approx_{ES_3} bad$ by (C2), even though $\{a, b\}$ is not enabled at $\emptyset$. □

5.4 L-Event Structures and Petri Nets

In this section we show how an L-event structure can be associated with every Petri net. The construction is based on the equivalence relation $\approx_{SFS_N}$ defined in Section 5.3 over the prime intervals $PI_{SFS_N} = \{\rho t \mid \rho t \in SFS_N\}$ associated with the set SFS_N of step firing sequences of a Petri net N. Note that SFS_N satisfies the conditions (L1) and (L2) from Section 5.3 which implies that $\approx_{SFS_N}$ can be defined. Since associating an L-event structure with a Petri net N is done on the basis of SFS_N rather than MFS_N, possible auto-concurrency in N is not taken into account. In this sense our event structure semantics is restricted. In Section 5.7 and Section 5.8 we will say more about this restriction.

In what follows we write PI_N, int_N, $\approx_N$, $\langle \rho t \rangle_N$, and $past_N$ rather than PI_{SFS_N}, int_{SFS_N}, $\approx_{SFS_N}$, $\langle \rho t \rangle_{\approx_N}$, and $past_{\approx_N}$, respectively.

Using these notions we first define for each Petri net N an L-event structure $nu(N)$. Then we prove that $nu(N)$ satisfies the unique occurrence property defined in Section 5.3 and is thus an UL-event structure.

Definition 5.4.1

Let N be a Petri net. Then $nu(N) = (E, C, \vdash)$ where

- $E = \{\langle \rho t \rangle_N \mid \rho t \in PI_N\}$
- $C = \{past_N(\rho) \mid \rho \in SFS_N\}$
- $\vdash \subseteq C \times P_F(E)$ is given by:

$$c \vdash u \Leftrightarrow \exists \rho v \in SFS_N.\, (past_N(\rho) = c \text{ and } u = \{\langle \rho t \rangle_N \mid t \in v\}).$$

□

Lemma 5.4.2

Let N be a Petri net. Then $nu(N) = (E, C, \vdash)$ is an L-event structure.

Proof.

Let $\emptyset \neq \hat{c} \in C$. Then there exists $\rho u \in SFS_N$ such that $u \neq \emptyset$ and $\hat{c} = past_N(\rho u)$. Let $t \in u$. Then $\rho(u-t)t \in SFS_N$. Hence $past_N(\rho(u-t)) \vdash \langle \rho(u-t)t \rangle_N$. By condition (C1) we have that $\rho t \approx_N \rho(u-t)t$. Then $\langle \rho t \rangle_N \notin past_N(\rho(u-t))$ by Lemma 5.3.5. Hence $past_N(\rho(u-t)) = past_N(\rho u) - \langle \rho t \rangle_N$ and thus $\hat{c} - \langle \rho t \rangle_N \vdash \langle \rho t \rangle_N$. This proves that $nu(N)$ satisfies (E0).

Since $\rho\emptyset \in SFS_N$ for all $\rho \in SFS_N$, we have that $\hat{c} \vdash \emptyset$, for all $\hat{c} \in C$, and so $nu(N)$ also satisfies (E1).

Let $\hat{c} \in C$ and $\hat{u} \in P_F(E)$ be such that $\hat{c} \vdash \hat{u}$. Let $\rho u \in SFS_N$ be such that $past_N(\rho) = \hat{c}$ and $\hat{u} = \{\langle \rho t \rangle_N \mid t \in u\}$. Then $\hat{c} \cap \hat{u} = \emptyset$ by Lemma 5.3.5. Now let $\hat{v} \subseteq \hat{u}$. Let $v \subseteq u$ be such that $\hat{v} = \{\langle \rho t \rangle_N \mid t \in v\}$. Then $\rho v(u-v) \in SFS_N$. Hence $\hat{c} \vdash \hat{v}$ and $\hat{c} \cup \hat{v} \vdash \{\langle \rho v t \rangle_N \mid t \in u - v\}$. For all $t \in u - v, \rho(v \cup t) \in SFS_N$ and so by condition (C1), $\rho t \approx_N \rho v t$. Therefore $\{\langle \rho v t \rangle_N \mid t \in u - v\} = \hat{u} - \hat{v}$. This proves that $nu(N)$ satisfies (E2). □

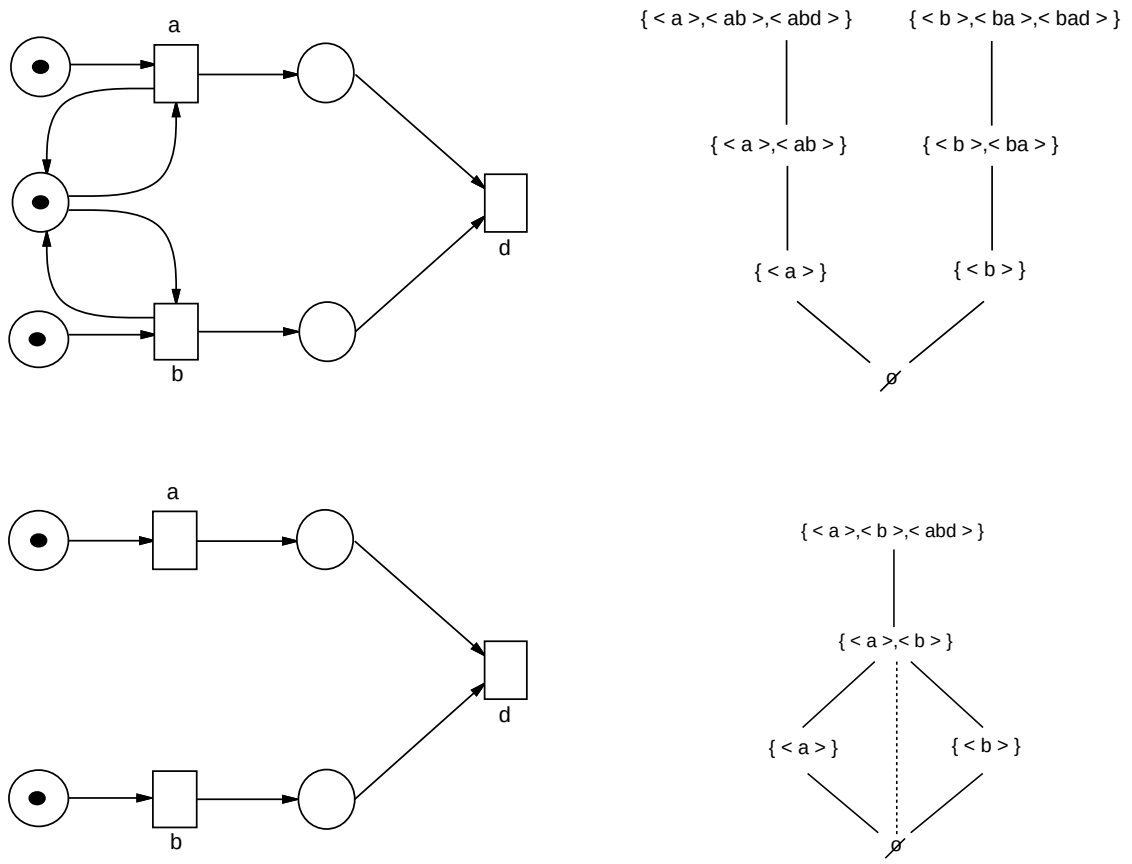


Figure 5.5: The 1-safe Petri nets N_3 and N_4 with their associated L-event structures

Example 5.4.3

In Figure 5.5 the 1-safe Petri nets N_3 and N_4 are depicted together with the L-event structures $nu(N_3)$ and $nu(N_4)$. The L-event structure $nu(N_3)$ has six events, for each of which there exists exactly one equivalence class of prime intervals (under $\approx_{nu(N_3)}$). Hence $nu(N_3)$ has the unique occurrence property. Also in $nu(N_4)$ there is a bijective correspondence between its set of events and its set of equivalence classes of prime intervals: both consist of three elements.

Let N_{10} be the Petri net depicted in Figure 5.4. The L-event structure $nu(N_{10})$ is ES_3 from Example 5.2.2 (where the unique equivalence class corresponding to each transition has been replaced by the transition itself). Thus, also $nu(N_{10})$ has the unique occurrence property. □

The L-event structures associated with the Petri nets in Example 5.4.3 all have the unique occurrence property. Now we turn to the proof that for any Petri net N, the L-event structure $nu(N) = (E, C, \vdash)$ has the unique occurrence property. First it is shown how the set of step firing sequences of $nu(N)$ can be derived from the set of step firing sequences of N by means of a function seq_N which associates with every step firing sequence of N a step sequence over E.

Define the function $seq_N : SFS_N \to (P_F(E))^+$ inductively by:

- $seq_N(\emptyset) = \emptyset$
- $seq_N(\rho u) = seq_N(\rho)\{\langle \rho t \rangle_N \mid t \in u\}$.

If the Petri net N is clear from the context, then we may omit the subscript $_N$ in seq_N.

Lemma 5.4.4
Let $N = (S, T, W, M_{in})$ be a Petri net. Then $seq(SFS_N) = SFS_{nu(N)}$.

Proof.
Let $nu(N) = (E, C, \vdash)$. Let $\rho \in SFS_N$. We prove that $seq(\rho) \in SFS_{nu(N)}$ and $cf(seq(\rho)) = past_N(\rho)$ by induction on $|\rho|$. If $\rho = \emptyset$ then this is clear, so assume that $\rho = \rho' u$ with $\rho' \in SFS_N$ and $\emptyset \neq u \in P_F(T)$. By the induction hypothesis $seq(\rho') \in SFS_{nu(N)}$ and $cf(seq(\rho')) = past_N(\rho')$. We also have, by the definition of $\vdash$, that $past_N(\rho') \vdash \hat{u}$ where $\hat{u} = \{\langle \rho' t \rangle_N \mid t \in u\}$. Hence $seq(\rho')\hat{u} \in SFS_{nu(N)}$ and $cf(seq(\rho')\hat{u}) = past_N(\rho') \cup \hat{u}$. Since $seq(\rho')\hat{u} = seq(\rho)$ and $past_N(\rho') \cup \hat{u} = past_N(\rho)$, we can now conclude that $seq(\rho) \in SFS_{nu(N)}$ and $cf(seq(\rho)) = past_N(\rho)$.

Now let $\hat{\rho} \in SFS_{nu(N)}$. We prove by induction on $|\hat{\rho}|$ that there exists $\rho \in SFS_N$ with $seq(\rho) = \hat{\rho}$ and $past_N(\rho) = alph(\hat{\rho})$. If $\hat{\rho} = \emptyset$ then $\rho = \emptyset$ is as required, so assume that $\hat{\rho} = \hat{\rho'}\hat{u}$ with $\hat{\rho'} \in SFS_{nu(N)}$ and $\emptyset \neq \hat{u} \in P_F(E)$. By the induction hypothesis there exists $\rho' \in SFS_N$ such that $seq(\rho') = \hat{\rho'}$ and $past_N(\rho') = alph(\hat{\rho'})$. Since $past_N(\rho') \vdash \hat{u}$ there exist $\rho_1 \in SFS_N$ and $u \in P_F(T)$ such that $\rho_1 u \in SFS_N$, $past_N(\rho_1) = past_N(\rho')$, and $\hat{u} = \{\langle \rho_1 t \rangle_N \mid t \in u\}$. From $past_N(\rho_1) = past_N(\rho')$ and Lemma 5.3.4(1) it easily follows that $num_t(\rho_1) = num_t(\rho')$ for all $t \in T$ and hence ρ_1 and ρ' lead to the same marking. Then we know from $\rho_1 u \in SFS_N$ that also $\rho' u \in SFS_N$. Moreover, $\langle \rho_1 t \rangle_N = \langle \rho' t \rangle_N$ for all $t \in u$ by condition (C2). Hence $seq(\rho' u) = seq(\rho')\{\langle \rho' t \rangle_N \mid t \in u\} = \hat{\rho'}\hat{u}$ and $past_N(\rho' u) = past_N(\rho') \cup \{\langle \rho' t \rangle_N \mid t \in u\} = alph(\hat{\rho'}) \cup \hat{u} = alph(\hat{\rho'}\hat{u})$. □

The above lemma allows us to characterize $int_{nu(N)}$, the function associating with each step firing sequence of $nu(N)$ its set of prime intervals, as follows.

Lemma 5.4.5
Let $N = (S, T, W, M_{in})$ be a Petri net and let $\rho \in SFS_N$. Then

$$int_{nu(N)}(seq(\rho)) = \{seq(\rho')\langle \rho' t \rangle_N \mid \rho' t \in int_N(\rho)\}.$$

Proof.
If $\rho = \emptyset$ then the claim trivially holds, so assume that $\rho = \rho_1 u$ with $\rho_1 \in SFS_N$ and $\emptyset \neq u \in P_F(T)$ and suppose that $int_{nu(N)}(seq(\rho_1)) = \{seq(\rho')\langle \rho' t \rangle_N \mid \rho' t \in int_N(\rho_1)\}$. Then $int_{nu(N)}(seq(\rho)) = int_{nu(N)}(seq(\rho_1)) \cup \{seq(\rho_1)\hat{t} \mid \hat{t} \in \{\langle \rho_1 t \rangle_N \mid t \in u\}\} = \{seq(\rho')\langle \rho' t \rangle_N \mid \rho' t \in int_N(\rho)\}$. □

Lemma 5.4.4 implies a close relationship between the prime intervals of a Petri net N and the prime intervals of $nu(N)$:

$$PI_{nu(N)} = \{seq(\rho)\langle\rho t\rangle_N \mid \rho t \in PI_N\}.$$

Using Lemma 5.4.4 and Lemma 5.4.5 it is shown next that there is also a strong correspondence between the equivalence classes of prime intervals under $\approx_N$ and $\approx_{nu(N)}$.

Lemma 5.4.6

Let N be a Petri net and let $\rho_1 t_1, \rho_2 t_2 \in PI_N$. Then

$$\rho_1 t_1 \approx_N \rho_2 t_2 \Leftrightarrow seq(\rho_1)\langle\rho_1 t_1\rangle_N \approx_{nu(N)} seq(\rho_2)\langle\rho_2 t_2\rangle_N.$$

Proof.

If $seq(\rho_1)\langle\rho_1 t_1\rangle_N \approx_{nu(N)} seq(\rho_2)\langle\rho_2 t_2\rangle_N$, then by Lemma 5.3.4(1) $\langle\rho_1 t_1\rangle_N = \langle\rho_2 t_2\rangle_N$.

In order to prove the implication in the other direction, assume that $\langle\rho_1 t_1\rangle_N = \langle\rho_2 t_2\rangle_N$. Define the equivalence relation $R \subseteq PI_N \times PI_N$ by: $\rho t\, R\, \rho' t'$ iff $seq(\rho)\langle\rho t\rangle_N \approx_{nu(N)} seq(\rho')\langle\rho' t'\rangle_N$. Suppose that R is SFS_N-consistent. Since $\approx_N$ is the least equivalence relation which is SFS_N-consistent it follows that $\approx_N \subseteq R$. Hence $\rho_1 t_1\, R\, \rho_2 t_2$ and thus, by the definition of R, $seq(\rho_1)\langle\rho_1 t_1\rangle_N \approx_{nu(N)} seq(\rho_2)\langle\rho_2 t_2\rangle_N$.

In order to prove that R satisfies (C1), suppose $\rho u \in SFS_N$ and $t \in u$. Since $\approx_N$ satisfies (C1), we have $\langle\rho t\rangle_N = \langle\rho(u-t)t\rangle_N$. We also have, by Lemma 5.4.4, that $seq(\rho u) \in SFS_{nu(N)}$. Combining this with $\approx_{nu(N)}$ satisfies (C1) leads to $seq(\rho)\langle\rho t\rangle_N \approx_{nu(N)} seq(\rho)(\hat{u} - \langle\rho t\rangle_N)\langle\rho t\rangle_N$ where $\hat{u} = \{\langle\rho t'\rangle_N \mid t' \in u\}$, because . Since $seq(\rho)(\hat{u} - \langle\rho t\rangle_N) = seq(\rho(u-t))$, we can now conclude by the definition of R that $\rho t\, R\, \rho(u-t)t$. This proves that R satisfies (C1).

Now suppose $\rho t, \rho' t \in PI_N$ are such that $past_R(\rho) = past_R(\rho')$. In order to prove that $\rho t\, R\, \rho' t$, we must show that $seq(\rho)\langle\rho t\rangle_N \approx_{nu(N)} seq(\rho')\langle\rho' t\rangle_N$. Because $\approx_{nu(N)}$ satisfies (C2), it suffices to prove that $past_{nu(N)}(seq(\rho)) = past_{nu(N)}(seq(\rho'))$ and $\langle\rho t\rangle_N = \langle\rho' t\rangle_N$.

First we prove that $past_{nu(N)}(seq(\rho)) = past_{nu(N)}(seq(\rho'))$. Suppose $\langle\hat{\rho}_1\hat{t}_1\rangle_{nu(N)} \in past_{nu(N)}(seq(\rho))$. Then there exists $\hat{\rho}_3\hat{t}_3 \in int(seq(\rho))$ such that $\langle\hat{\rho}_1\hat{t}_1\rangle_{nu(N)} = \langle\hat{\rho}_3\hat{t}_3\rangle_{nu(N)}$. By Lemma 5.4.5 there exists $\rho_3 t_3 \in int(\rho)$ such that $\hat{\rho}_3\hat{t}_3 = seq(\rho_3)\langle\rho_3 t_3\rangle_N$. Then $\langle\rho_3 t_3\rangle_R \in past_R(\rho) = past_R(\rho')$. Hence there exists $\rho_4 t_4 \in int(\rho')$ such that $\langle\rho_3 t_3\rangle_R = \langle\rho_4 t_4\rangle_R$. Then, again by Lemma 5.4.5, $seq(\rho_4)\langle\rho_4 t_4\rangle_N \in int(seq(\rho'))$. Moreover, $\hat{\rho}_3\hat{t}_3 \approx_{nu(N)} seq(\rho_4)\langle\rho_4 t_4\rangle_N$ by the definition of R. Hence $\langle\hat{\rho}_1\hat{t}_1\rangle_{nu(N)} = \langle seq(\rho_4)\langle\rho_4 t_4\rangle_N\rangle_{nu(N)} \in past_{nu(N)}(seq(\rho'))$. This proves that $past_{nu(N)}(seq(\rho)) \subseteq past_{nu(N)}(seq(\rho'))$. By a symmetric argument we can show that $past_{nu(N)}(seq(\rho')) \subseteq past_{nu(N)}(seq(\rho))$ and thus $past_{nu(N)}(seq(\rho)) = past_{nu(N)}(seq(\rho'))$.

In order to prove that $\langle\rho t\rangle_N = \langle\rho' t\rangle_N$, it suffices to prove that $past_N(\rho) = past_N(\rho')$ because $\approx_N$ satisfies (C2). Let $\langle\rho_3 t_3\rangle_N \in past_N(\rho)$. Then there exists $\rho_4 t_4 \in int(\rho)$ such that $\langle\rho_3 t_3\rangle_N = \langle\rho_4 t_4\rangle_N$. By Lemma 5.4.5 we now have that $\hat{\rho}_4\hat{t}_4 \in int(seq(\rho))$ where $\hat{\rho}_4 = seq(\rho_4)$ and $\hat{t}_4 = \langle\rho_4 t_4\rangle_N$. Hence $\langle\hat{\rho}_4\hat{t}_4\rangle_{nu(N)} \in past_{nu(N)}(seq(\rho)) = past_{nu(N)}(seq(\rho'))$. Then there exists $\hat{\rho}_5\hat{t}_5 \in int(seq(\rho'))$ such that $\langle\hat{\rho}_4\hat{t}_4\rangle_{nu(N)} = \langle\hat{\rho}_5\hat{t}_5\rangle_{nu(N)}$. By Lemma 5.3.4(1), $\hat{t}_4 = \hat{t}_5$. By Lemma 5.4.5 there exists $\rho_5 t_5 \in int(\rho')$ such that $\hat{\rho}_5 = seq(\rho_5)$ and $\hat{t}_5 = \langle\rho_5 t_5\rangle_N$. Then $\hat{t}_5 \in past_N(\rho')$, and so $\langle\rho_3 t_3\rangle_N =$

$\hat{t}_4 = \hat{t}_5 \in past_N(\rho')$. This proves that $past_N(\rho) \subseteq past_N(\rho')$. Similarly we have that $past_N(\rho') \subseteq past_N(\rho)$ and thus $past_N(\rho) = past_N(\rho')$.

This finishes the proof that R satisfies (C2). We can conclude that $seq(\rho_1)\langle\rho_1 t_1\rangle_N \approx_{nu(N)} seq(\rho_2)\langle\rho_2 t_2\rangle_N$. □

This leads to the desired result that for each Petri net N, $nu(N)$ has the unique occurrence property.

Theorem 5.4.7

Let $N = (S, T, W, M_{in})$ be a Petri net. Then $nu(N)$ is an UL-event structure.

Proof.

By Lemma 5.4.2, $nu(N)$ is an L-event structure. We must verify that $nu(N)$ satisfies the conditions (U1) and (U2) specified in the definition of the unique occurrence property.

Let $nu(N) = (E, C, \vdash)$. If $\langle\rho t\rangle_N \in E$ then $\rho t \in SFS_N$ and hence $past_N(\rho) \vdash \langle\rho t\rangle_N$. Hence $nu(N)$ satisfies (U1). Now in order to prove (U2), let $\hat{\rho}_1\hat{t}_1, \hat{\rho}_2\hat{t}_2 \in PI_{nu(N)}$ be such that $\hat{t}_1 = \hat{t}_2$. By Lemma 5.4.4 there exist $\rho_1, \rho_2 \in SFS_N$ and $t_1, t_2 \in T$ such that $\rho_1 t_1, \rho_2 t_2 \in SFS_N$, $\hat{\rho}_1 = seq(\rho_1)$, $\hat{\rho}_2 = seq(\rho_2)$, $\hat{t}_1 = \langle\rho_1 t_1\rangle_N$, and $\hat{t}_2 = \langle\rho_2 t_2\rangle_N$. Since $\hat{t}_1 = \hat{t}_2$ we then have by Lemma 5.4.6, that $\hat{\rho}_1\hat{t}_1 \approx_{nu(N)} \hat{\rho}_2\hat{t}_2$. □

One of the reasons for giving an event structure semantics for Petri nets is that we want to distinguish between different occurrences of transitions in a Petri net. The fact proved in Theorem 5.4.7 that the L-event structures yielded by nu have the unique occurrence property ensures that in the L-event structure associated with a Petri net the transition occurrences cannot be distinguished any further.

To conclude this section we give the event structure semantics for one of our running examples.

Example 5.4.8

In Figure 5.6 the UL-event structure associated with the Petri net N_1 is depicted. After each event representing an occurrence of a, both an event representing the occurrence of b and an event representing the occurrence of c are enabled. Whereas all events representing occurrences of b are different, due to conflicts in the Petri net, the events representing occurrences of c are the same. □

5.5 PN-Event Structures

In [66] it is not only shown how to associate a prime event structure with each 1-safe Petri net, but also a map from prime event structures to 1-safe Petri nets is given. Our aim is to lift this construction also here; in other words, to set up a map from UL-event structures to Petri nets. It turns out that the construction we have in mind works for *all* L-event structures. Hence we construct a map from L-event structures to Petri nets.

The map from L-event structures to Petri nets is used in the proof of the main result of this section which states that all UL-event structures can be obtained (up

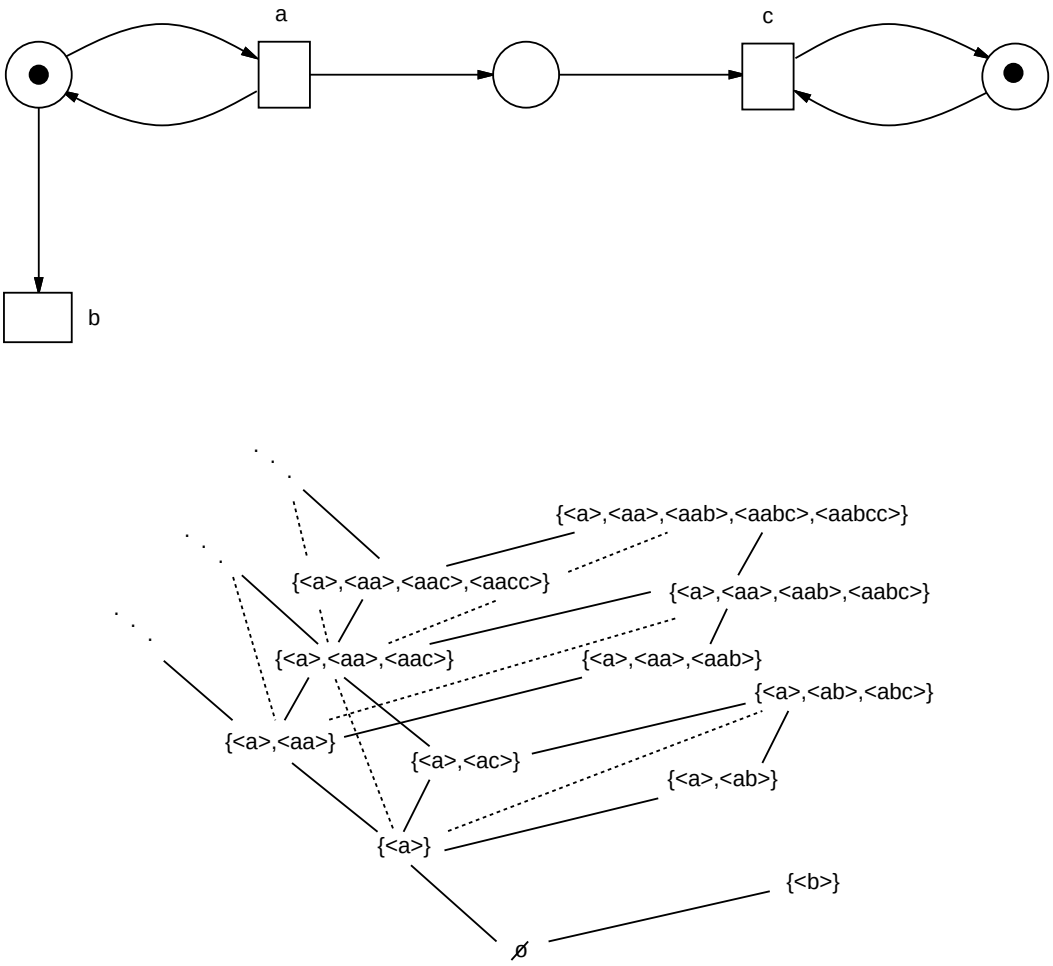


Figure 5.6: The Petri net N_1 with its associated L-event structure

to isomorphism) from Petri nets through the map nu. Hence the class of L-event structures yielded by Petri nets is exactly the class of UL-event structures. Thus whereas for the PN-transition systems and the PN-trace languages regional axioms are needed for their characterization, for L-event structures the unique occurrence property suffices to characterize the *PN-event structures*.

Definition 5.5.1

An L-event structure ES is a *PN-event structure* if there exists a Petri net N such that $ES \equiv nu(N)$. □

Given a prime event structure $(E, \leq, \#)$, the causality relation $\leq$ and the conflict relation $\#$ make it possible in [66] to quickly manufacture a suitable set of places. It is then easy to associate, in a canonical way, a 1-safe Petri net with each prime event structure. In the setting of general Petri nets, it is far from clear what causality, concurrency, and conflict could mean. Fortunately, we can use again the regional

construction as defined in Section 3.2 by representing L-event structures as multiset transition systems.

Let $ES = (E, C, \vdash)$ be an L-event structure. Then define the multiset transition system $et(ES) = (C, E, \longrightarrow_{ES}, \emptyset)$ where $(C, E, \longrightarrow_{ES})$ is the multiset transition diagram associated with ES as defined in Section 5.2. Recall that $\longrightarrow_{ES} \subseteq C \times P_F(E) \times C$ is given by:

$$c \xrightarrow{u}_{ES} c' \Leftrightarrow (c \vdash u \text{ and } c' = c \cup u).$$

Thus we can speak now of the regions of ES. So a region of ES is a function $r : C \cup E \to \mathbf{N} \cup (\mathbf{N} \times \mathbf{N})$ satisfying the following conditions.

(1) $\forall c \in C.\, r(c) \in \mathbf{N}$ and $\forall e \in E.\, r(e) \in \mathbf{N} \times \mathbf{N}$.

(2) $c \vdash u \Rightarrow (r(c) \geq \sum_{e \in u} {}^{r}e$ and $r(c \cup u) = r(c) + \sum_{e \in u}(e^{r} - {}^{r}e))$.

From now on we write $\mathcal{R}_{ES}$ instead of $\mathcal{R}_{et(ES)}$.

Definition 5.5.2

The map en from L-event structures to Petri nets is given by $en = tn \circ et$. Hence $en(ES) = (\mathcal{R}_{ES}, E, W, M_{in})$ where

- $W : (\mathcal{R}_{ES} \times E) \cup (E \times \mathcal{R}_{ES}) \to \mathbf{N}$ is such that

$$\forall r \in \mathcal{R}_{ES}.\, \forall e \in E.\, (W(r, e) = {}^{r}e \text{ and } W(e, r) = e^{r})$$

- $M_{in} : \mathcal{R}_{ES} \to \mathbf{N}$ is such that

$$\forall r \in \mathcal{R}_{ES}.\, M_{in}(r) = r(\emptyset).$$

□

The following lemma shows that $en(ES)$ has the same step firing sequences as ES. Moreover, it turns out that $MFS_{en(ES)} = SFS_{en(ES)}$ and so $en(ES)$ is a co-safe Petri net. While it easily follows that $SFS_{ES} \subseteq SFS_{en(ES)}$, the converse inclusion requires a more complicated proof showing that ES has enough regions to prevent the existence of "wrong" step firing sequences in $SFS_{en(ES)}$.

Lemma 5.5.3

Let $ES = (E, C, \vdash)$ be an L-event structure. Then $SFS_{ES} = MFS_{en(ES)} = SFS_{en(ES)}$.

Proof.

Let $en(ES) = (\mathcal{R}_{ES}, E, W, M_{in})$. Let for each $e \in E$ the function $r_e : C \cup E \to \mathbf{N} \cup (\mathbf{N} \times \mathbf{N})$ be given by:

(1) $\forall e' \in E.\, r_e(e') = \begin{cases} (1,1) & \text{if } e' = e \\ (0,0) & \text{otherwise} \end{cases}$

(2) $\forall c \in C.\, r_e(c) = 1$.

Then each r_e is a non-trivial region of ES, and so it is clear that $MFS_{en(ES)} = SFS_{en(ES)}$.

Suppose $\rho \in SFS_{ES}$. Then $\emptyset \xrightarrow{\rho}_{ES}$. This implies by Lemma 3.2.7(2) that $\rho \in MFS_{en(ES)} = SFS_{en(ES)}$.

Conversely, suppose that $\rho \in SFS_{en(ES)}$. We prove by induction on $|\rho|$ that $\rho \in SFS_{ES}$ and, for all $r \in \mathcal{R}_{ES}$, $M_\rho(r) = r(alph(\rho))$. If $\rho = \emptyset$ then this is clear, so assume that $\rho = \rho' u$ with $\rho' \in SFS_{en(ES)}$ and $\emptyset \neq u \in P_F(E)$. By the induction hypothesis $\rho' \in SFS_{ES}$ and, for all $r \in \mathcal{R}_{ES}$, $M_{\rho'}(r) = r(alph(\rho'))$. We first prove that $alph(\rho') \cap u = \emptyset$.

Suppose $e \in alph(\rho')$. Then define $r\langle e\rangle : C \cup E \to \mathbf{N} \cup (\mathbf{N} \times \mathbf{N})$ as follows.

$$(1)\ \forall e' \in E.\, r\langle e\rangle(e') = \begin{cases} (1,0) & \text{if } e' = e \\ (0,0) & \text{otherwise.} \end{cases}$$

$$(2)\ \forall c \in C.\, r\langle e\rangle(c) = \begin{cases} 0 & \text{if } e \in c \\ 1 & \text{otherwise.} \end{cases}$$

Claim 1. $r\langle e\rangle \in \mathcal{R}_{ES}$.

Let us assume that Claim 1 holds. Then we have $M_{\rho'}(r\langle e\rangle) = r\langle e\rangle(alph(\rho')) = 0$. In addition we know that $W(r\langle e\rangle, e) = 1$ and, because $\rho' u \in SFS_{en(ES)}$, we also know that $M_{\rho'}(r\langle e\rangle) \geq \sum_{e' \in u} W(r\langle e\rangle, e')$. All this leads to the conclusion that $e \notin u$. This proves that $alph(\rho') \cap u = \emptyset$.

Now we observe that $\rho = \rho' u \in SFS_{ES}$ if $alph(\rho') \vdash u$. So denote $c = alph(\rho')$ and assume that $c \vdash u$ does not hold. This leads to a contradiction as we show next.

Define $r\langle u, c\rangle : C \cup E \to \mathbf{N} \cup (\mathbf{N} \times \mathbf{N})$ as follows.

$$(1)\ \forall e \in E.\, r\langle u,c\rangle(e) = \begin{cases} (1,0) & \text{if } e \in c \\ (1,1) & \text{if } e \in u \\ (0,1) & \text{otherwise.} \end{cases}$$

$$(2)\ \forall c' \in C.\, r\langle u,c\rangle(c') = |c| + |u| - 1 + \textstyle\sum_{e \in c'} (e^{r\langle u,c\rangle} - {}^{r\langle u,c\rangle}e).$$

Claim 2. $r\langle u, c\rangle \in \mathcal{R}_{ES}$.

Assume that Claim 2 holds. Then $M_{\rho'}(r\langle u,c\rangle) = r\langle u,c\rangle(c) = |u| - 1 < |u| = \sum_{e \in u} {}^{r\langle u,c\rangle}e = \sum_{e \in u} W(r\langle u,c\rangle, e)$, a contradiction with $\rho' u \in SFS_{en(ES)}$. Thus $c \vdash u$ and hence $\rho = \rho' u \in SFS_{ES}$. Moreover, $r(alph(\rho)) = r(c \cup u) = r(c) + \sum_{e \in u}(e^r - {}^r e) = M_{\rho'}(r) + \sum_{e \in u}(W(e,r) - W(r,e)) = M_\rho(r)$ for all $r \in \mathcal{R}_{ES}$.

Thus if we can prove Claim 1 and Claim 2, then we can conclude that $SFS_{ES} = SFS_{en(ES)}$.

Proof of Claim 1.

To simplify the notation we write r instead of $r\langle e\rangle$. Suppose $c' \vdash v$. Since $c' \cap v = \emptyset$ by (E2) we then have that $r(c' \cup v) = r(c') - |v \cap e| = r(c') + \sum_{e' \in v}(e'^r - {}^r e')$ and $r(c') = r(c' \cup v) + |v \cap e| \geq |v \cap e| = \sum_{e' \in v} {}^r e'$. Hence r is a region of ES which is clearly non-trivial. This proves Claim 1.

Proof of Claim 2.

In order to simplify the notation, we write r instead of $r\langle u, c\rangle$ in this proof.

Suppose $c' \in C$ and $v \in P_F(E)$ are such that $c' \vdash v$. Since $c' \cap v = \emptyset$ by (E2) we immediately have that $r(c' \cup v) = r(c') + \sum_{e \in v}(e^r - {}^r e)$. Now we must prove that $r(c') \geq \sum_{e \in v} {}^r e$.

Let $n = |v \cap (c \cup u)| = \sum_{e \in v} {}^r e$. Then we must prove that $r(c') \geq n$. Set $k = |c' \cap u|$ and $j = |c' \cap c|$ and $m = |c' \cap (E - (c \cup u))|$. Since $c \cap u = \emptyset$ and $c' \cap v = \emptyset$ it follows that $n \leq |c| + |u| - k - j$. Moreover, by the definition of r, it is clear that $r(c') = |c| + |u| - 1 + k + m - k - j = |c| + |u| - 1 + m - j$. Hence if $m + k \geq 1$ we are done. Therefore we assume in the rest of the proof that $m = k = 0$. In other words, we assume that $c' \subseteq c$. This leads to the equation $r(c') = |c| + |u| - 1 - |c'|$. On the other hand, $n \leq |c| + |u| - |c'|$. If $n < |c| + |u| - |c'|$ then we at once get $r(c') \geq n$. We now wish to argue that $n = |c| + |u| - |c'|$ leads to a contradiction.

To see this, suppose that $n = |c| + |u| - |c'|$. Let $v_1 = v \cap c$ and $v_2 = v \cap u$. Then from $c' \cap v = \emptyset$ and $c' \subseteq c$ it follows that $v_1 = c - c'$ and $v_2 = u$. Since $c' \vdash v$ we also have that $c' \vdash (v_1 \cup v_2)$ by (E2). Again by (E2) we now know that $(c' \cup v_1) \vdash v_2$. Since $c' \cup v_1 = c$ and $v_2 = u$ this leads to a contradiction. This proves that $n = |c| + |u| - |c'|$ is not possible, so $r(c') \geq n$.

This proves that r is a region of ES. Since $u \neq \emptyset$, r is also non-trivial. This finishes the proof of Claim 2. □

From the proof of the above lemma it follows that $en(ES)$ is not just a co-safe Petri net. In fact $en(ES)$ has enough places to ensure that it is a *locally sequential* Petri net.

A locally sequential Petri net is a Petri net $N = (S, T, W, M_{in})$ where for each $t \in T$ there exists a "private" place $s_t \in S$ such that $M_{in}(s_t) = 1$ and, for each $x \in T$, $W(s_t, x) = W(x, s_t) = 1$ if $x = t$ and $W(s_t, x) = W(x, s_t) = 0$ otherwise.

Thus in a locally sequential Petri net co-safety is guaranteed by purely structural means.

Recall that our main aim is to associate a Petri net with every UL-event structure. It turns out that our map en (which acts on all L-event structures), when restricted to UL-event structures, fits in very well with the map nu from Petri nets to UL-event structures given in Section 5.4.

Let $ES = (E, C, \vdash)$ be an UL-event structure with $nu(en(ES)) = (\hat{E}, \hat{C}, \hat{\vdash})$. Define $\upsilon_{ES} : E \to \hat{E}$ as follows. Let $e \in E$. By the unique occurrence property there exists a unique equivalence class $\langle \rho e \rangle_{ES}$. Now let

$$\upsilon_{ES}(e) = \langle \rho e \rangle_{en(ES)}.$$

By Lemma 5.5.3, $SFS_{ES} \subseteq SFS_{en(ES)}$. Hence by Lemma 5.3.4(2), $\upsilon_{ES}(e)$ is well-defined.

Lemma 5.5.4

Let ES be an UL-event structure. Then υ_{ES} an LES-isomorphism from ES to $nu(en(ES))$ and so $ES \equiv nu(en(ES))$.

Proof.

Let $ES = (E, C, \vdash)$ and $nu(en(ES)) = (\hat{E}, \hat{C}, \hat{\vdash})$ and let $c \in C$ and $u \in P_F(E)$.

Suppose $c \vdash u$. Let $\rho \in SFS_{ES}$ be such that $alph(\rho) = c$. Then $\rho u \in SFS_{ES}$ and hence $\rho u \in SFS_{en(ES)}$ by Lemma 5.5.3. This implies by the definition of nu that $past_{en(ES)}(\rho) \hat{\vdash} \hat{u}$ where $\hat{u} = \{\langle \rho e \rangle_{en(ES)} \mid e \in u\}$. In order to prove that $v_{ES}(c) \hat{\vdash} v_{ES}(u)$ we must prove that $v_{ES}(c) = past_{en(ES)}(\rho)$ and $v_{ES}(u) = \hat{u}$.

Suppose $e_1 \in c$ with $\rho_1 e_1 \in PI_{ES}$ such that $v_{ES}(e_1) = \langle \rho_1 e_1 \rangle_{en(ES)}$. From $e_1 \in alph(\rho)$ it follows that there exists $\rho_1' e_1 \in int_{ES}(\rho) = int_{en(ES)}(\rho)$. Moreover, by the unique occurrence property $\langle \rho_1 e_1 \rangle_{ES} = \langle \rho_1' e_1 \rangle_{ES}$ and hence, by Lemma 5.3.4(1) and Lemma 5.5.3, also $\langle \rho_1 e_1 \rangle_{en(ES)} = \langle \rho_1' e_1 \rangle_{en(ES)}$. Since $\langle \rho_1' e_1 \rangle_{en(ES)} \in past_{en(ES)}(\rho)$, this proves that $v_{ES}(e_1) \in past_{en(ES)}(\rho)$.

Now suppose $\langle \rho_1 e_1 \rangle_{en(ES)} \in past_{en(ES)}(\rho)$. Then there exists $\rho_1' e_1 \in int_{en(ES)}(\rho) = int_{ES}(\rho)$ such that $\langle \rho_1 e_1 \rangle_{en(ES)} = \langle \rho_1' e_1 \rangle_{en(ES)}$. Hence $e_1 \in alph(\rho) = c$ and $v_{ES}(e_1) = \langle \rho_1' e_1 \rangle_{en(ES)}$. This proves that $past_{en(ES)}(\rho) \subseteq v_{ES}(c)$ and hence $v_{ES}(c) = past_{en(ES)}(\rho)$. It easily follows that $v_{ES}(u) = \hat{u}$. Hence $v_{ES}(c) \hat{\vdash} v_{ES}(u)$. This proves that v_{ES} is an LES-morphism from ES to $nu(en(ES))$.

In order to prove that v_{ES} is an LES-isomorphism, suppose $v_{ES}(c) \hat{\vdash} v_{ES}(u)$. Then there exists $\rho v \in SFS_{en(ES)}$ such that $v_{ES}(c) = past_{en(ES)}(\rho)$ and $v_{ES}(u) = \{\langle \rho e \rangle_{en(ES)} \mid e \in v\}$. This implies that $c = alph(\rho)$ and $u = v$. Moreover, $\rho v \in SFS_{ES}$ by Lemma 5.5.3 and hence $c \vdash u$. Since v_{ES} is a bijection, we can conclude that v_{ES} is an LES-isomorphism. □

From Lemma 5.3.8, Theorem 5.4.7, and Lemma 5.5.4, we now obtain the following characterization of PN-event structures.

Theorem 5.5.5

An L-event structure is a PN-event structure iff it is an UL-event structure. □

Example 5.5.6

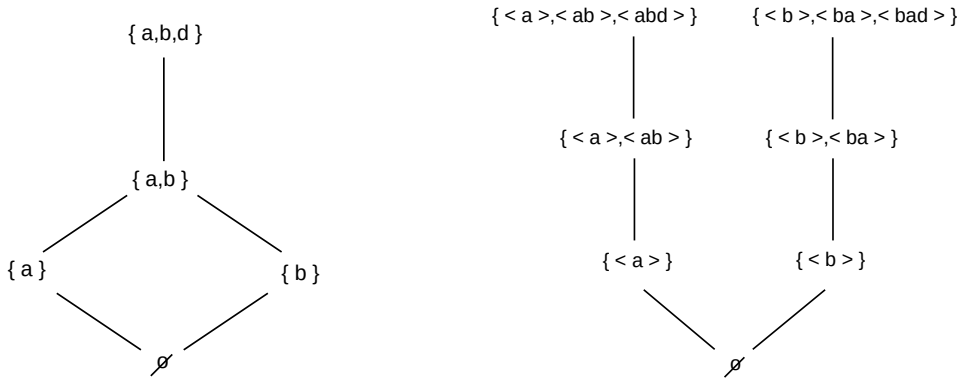


Figure 5.7: The L-event structures ES_6 and $nu(en(ES_6))$

Let ES_6 be the first L-event structure depicted in Figure 5.7. This L-event structure does not have the unique occurrence property and is hence not a PN-event structure.

The Petri net $en(ES_6)$ is the saturated version of the Petri net N_3 depicted in Figure 2.3. The L-event structure $nu(en(ES_6))$ is the second L-event structure depicted in Figure 5.7 (see also Example 5.4.3). Even though ES_6 has the same set of step firing sequences as $en(ES_6)$, the two L-event structures ES_6 and $nu(en(ES_6))$ are not isomorphic, because the event structure semantics for $en(ES_6)$ distinguishes more events than there are present in ES_6. □

5.6 L-Event Structures and 1-Safe Petri Nets

This section is devoted to the investigation of the relationship between the prime event structure semantics for 1-safe Petri nets as given in Definition 5.1.1 and our proposed L-event structure semantics for general Petri nets as given in Definition 5.4.1.

First it is shown how prime event structures can be viewed as L-event structures. Then it is shown as the main result of this section that the L-event structure semantics for Petri nets is a strictly conservative extension of the prime event structure semantics for 1-safe Petri nets: the L-event structure semantics when restricted to 1-safe Petri nets agrees completely (up to isomorphism) with the prime event structure semantics.

A prime event structure P is viewed as an L-event structure $pu(P)$ the configurations of which are the finite configurations of P. The enabling relation is given by the diamonds in the configuration structure of P.

Thus define $pu(P) = (E, FC_P, \vdash)$ where $\vdash \subseteq FC_P \times P_F(E)$ is given by:

$$c \vdash u \Leftrightarrow c \cap u = \emptyset \text{ and } \forall v \subseteq u.\, c \cup v \in FC_P.$$

Lemma 5.6.1

Let $P = (E, \leq, \#)$ be a prime event structure. Then $pu(P) = (E, FC_P, \vdash)$ is an L-event structure.

Proof.

In order to prove that $pu(P)$ satisfies (E0), let $\emptyset \neq c \in FC_P$. Let $e \in c$ be a maximal event in c in the sense that for all $e' \in c$, $e \leq e'$ implies that $e = e'$. Then $c - e \in FC_P$ and hence $c - e \vdash e$. This proves that $pu(P)$ satisfies (E0). From the definition of $pu(P)$ it easily follows that $pu(P)$ satisfies (E1) and (E2). □

In Chapter 6 this result will be strengthened by showing that the L-event structures obtained via pu all have the unique occurrence property (hence the name pu rather than pe). In Chapter 6 we will then also investigate the relationship between prime event structures and UL-event structures in a categorical framework.

For the moment we use the map pu only for showing that for a 1-safe Petri net its prime event structure semantics agrees with its L-event structure semantics via pu: for every 1-safe Petri net N, $nu(N) \equiv pu(sp(N))$.

Let N be a 1-safe Petri net, $nu(N) = (E, C, \vdash)$, and $pu(sp(N)) = (\hat{E}, FC_{sp(N)}, \hat{\vdash})$. Recall that the events in $\hat{E}$ are equivalence classes under $\sim_N$ of sequential prime intervals of N and the events in E are equivalence classes under $\approx_N$ of prime intervals of N. The LES-isomorphism $\lambda_N : \hat{E} \to E$ that will be used in the proof that $nu(N) \equiv pu(sp(N))$ maps each event $\langle \rho t \rangle_{\sim_N} \in \hat{E}$ to the event $\langle \rho t \rangle_{\approx_N} \in E$.

In order to show that λ_N is well-defined, it must be proved that equivalent sequential prime intervals under $\sim_N$ are also equivalent under $\approx_N$, that is:

$$\forall \rho t, \rho' t' \in sPI. (\rho t \sim_N \rho' t' \Rightarrow \rho t \approx_N \rho' t').$$

After having shown this, it must be proved (in order for this map to be an LES-isomorphism) that λ_N is a bijection. For proving that λ_N is surjective it is sufficient to prove that each event in E has a sequential representative:

$$\forall \rho t \in PI. \exists \rho' t' \in sPI. \rho t \approx_N \rho' t'.$$

For proving that λ_N is injective, it must be proved that equivalent sequential prime intervals under $\approx_N$ are also equivalent under $\sim_N$:

$$\forall \rho t, \rho' t' \in sPI. (\rho t \approx_N \rho' t' \Rightarrow \rho t \sim_N \rho' t').$$

Once it is proved that λ_N is a bijection, the proof that λ_N is an LES-isomorphism is fairly easy.

Now we turn to the proof that equivalence under $\sim_N$ is preserved. To prove this we need the following lemma.

Lemma 5.6.2

Let N be a 1-safe Petri net and let $\rho, \rho' \in FS$ be such that $\rho \doteq \rho'$. Then $past_N(\rho) = past_N(\rho')$.

Proof.

Suppose $\rho = \rho_1 t t' \rho_2$, $\rho' = \rho_1 t' t \rho_2$, and $(t, t') \in Ind_N$. Then by Lemma 2.1.12 also $\rho_1\{t, t'\} \in SFS$. Hence $\rho_1 t' t \approx \rho_1 t$ because $\approx$ satisfies (C1). Similarly, $\rho_1 t t' \approx \rho_1 t'$. Hence $past_N(\rho_1 t t') = past_N(\rho_1 t' t)$. Now it easily follows that also $past_N(\rho) = past_N(\rho')$ because $\approx$ satisfies (C2). □

Lemma 5.6.3

Let $N = (S, T, W, M_{in})$ be a 1-safe Petri net and let $\rho t, \rho' t' \in sPI$ be such that $\rho t \sim \rho' t'$. Then $\rho t \approx \rho' t'$.

Proof.

It is sufficient to prove that $\approx$ satisfies (S1) and (S2) because $\sim$ is the least equivalence relation satisfying (S1) and (S2). The required result then follows from $\sim \subseteq \approx$.

In order to prove that $\approx$ satisfies (S1), suppose $\rho_1 t_2 t_1 \in FS$ and $(t_1, t_2) \in Ind_N$. Then by Lemma 2.1.10, $\rho_1\{t_1, t_2\} \in SFS$. Since $\approx$ satisfies (C1) we then have that $\rho_1 t_1 \approx \rho_2 t_2 t_1$ which proves that $\approx$ satisfies (S1).

Now in order to prove that $\approx$ satisfies (S2), suppose $\rho_1 t_1, \rho_1' t_1 \in sPI$ are such that $\rho_1 \simeq \rho_1'$. Then by repeatedly applying Lemma 5.6.2, $past_N(\rho_1) = past_N(\rho')$. This now implies that $\rho_1 t_1 \approx \rho_1' t_1$ because $\approx$ satisfies (C2). Hence $\approx$ satisfies (S2). □

Given a 1-safe Petri net N with $nu(N) = (E, C, \vdash)$ and $pu(sp(N)) = (\hat{E}, FC_{sp(N)}, \hat{\vdash})$, by the above lemma the map $\lambda_N : \hat{E} \to E$ with $\lambda_N(\langle \rho t \rangle_{\sim_N}) = (\langle \rho t \rangle_{\approx_N})$ is well-defined.

We now turn to the proof that λ_N is surjective.

Lemma 5.6.4

Let N be a Petri net and let $\rho t \in PI$. Then

$$\exists \rho' t \in sPI.\, \rho t \approx \rho' t.$$

Proof.

Suppose $\rho = \rho_1 u \rho_2$ with $|u| > 1$ and $t' \in u$. Then $\rho_1(u - t')t'\rho_2 t \in PI$. Because $\rho_1 u \in SFS$ and $\approx$ satisfies (C1), $\rho_1 t' \approx \rho_1(u - t')t'$. This implies that $past_N(\rho_1 u) = past_N(\rho_1(u - t')t')$. Since $\approx$ satisfies (C2), it easily follows that $past_N(\rho_1 u \rho_2) = past_N(\rho_1(u - t')t'\rho_2)$. Again by (C2), we then also have that $\rho_1 u \rho_2 t \approx \rho_1(u - t')t'\rho_2 t$.

Repeatedly applying the above then yields the required $\rho' t \in sPI_N$ with $\rho t \approx \rho' t$. □

Hence each event $\langle \rho t \rangle_{\approx_N}$ in E has a representative $\rho' t$ in sPI which proves that λ_N is surjective.

To prove that λ_N is a bijection, it is now sufficient to prove that λ_N is injective. This is the most complicated step in our proof that λ_N is an LES-isomorphism.

We have to show that sequential prime intervals which are equivalent under $\approx_N$, are also equivalent under $\sim_N$. This is done by first lifting the equivalence relation $\sim_N$ over sequential prime intervals to an equivalence relation $\hat{\sim}_N$ over arbitrary prime intervals. Then we show that prime intervals which are equivalent under $\approx_N$ are also equivalent under $\hat{\sim}_N$:

$$\forall \rho t, \rho' t' \in PI.\, (\rho t \approx_N \rho' t' \Rightarrow \rho t \,\hat{\sim}_N\, \rho' t').$$

After having done this, the injectivity of λ_N follows by proving that sequential prime intervals which are equivalent under $\hat{\sim}_N$ are also equivalent under $\sim_N$:

$$\forall \rho t, \rho' t' \in sPI.\, (\rho t \,\hat{\sim}_N\, \rho' t' \Rightarrow \rho t \sim_N \rho' t').$$

Thus, for all $\rho t, \rho' t' \in sPI$, if $\rho t \not\sim_N \rho' t'$, then $\lambda_N(\langle \rho t \rangle_{\sim_N}) = \langle \rho t \rangle_{\approx_N} \neq \langle \rho' t' \rangle_{\approx_N} = \lambda_N(\langle \rho' t' \rangle_{\sim_N})$.

Let N be a 1-safe Petri net. The equivalence relation $\sim_N$ over sequential prime intervals is lifted to an equivalence relation over arbitrary prime intervals by replacing (S1) by condition (C1) and by replacing in (S2) the equivalence relation $\simeq_N$ by $\hat{\simeq}_N$, the extension of M-trace equivalence to step sequences defined in Section 4.6.

So define $\hat{\sim}_N \subseteq PI \times PI$ as the least equivalence relation which satisfies the following conditions (C1') and (C2').

(C1') $(\rho u \in SFS \text{ and } t \in u) \Rightarrow \rho t \,\hat{\sim}_N\, \rho(u - t)t.$

(C2') $(\rho t, \rho' t \in PI \text{ and } \rho \,\hat{\simeq}_N\, \rho') \Rightarrow \rho t \,\hat{\sim}_N\, \rho' t.$

If N is clear from the context then we may omit the subscript $_N$ in $\hat{\sim}_N$.

In order to be able to prove that $\approx_N \subseteq \hat{\sim}_N$ we first need the following two lemmas with properties of $\hat{\sim}_N$.

The first lemma states that prime intervals which are equivalent under $\hat{\sim}_N$ are associated with the same transition and can only differ in the number of occurrences of transitions which are independent of this transition.

Lemma 5.6.5

Let $N = (S, T, W, M_{in})$ be a 1-safe Petri net and let $\rho t, \rho' t' \in PI$ be such that $\rho t \hat{\sim} \rho' t'$. Then

$$t = t' \text{ and } \forall t'' \in T.\,((t, t'') \notin Ind_N \Rightarrow num_{t''}(\rho) = num_{t''}(\rho')).$$

Proof.

Define the equivalence relation $R \subseteq PI \times PI$ by: $\rho_1 t_1 \, R \, \rho_2 t_2$ iff $(t_1 = t_2$ and $\forall t'' \in T.\,((t_1, t'') \notin Ind_N \Rightarrow num_{t''}(\rho_1) = num_{t''}(\rho_2)))$. If R satisfies (C1') and (C2'), then $\hat{\sim} \subseteq R$, because $\hat{\sim}$ is the least equivalence relation satisfying (C1') and (C2'). This would imply that $\rho t \, R \, \rho' t'$ from which the required result follows. Thus it is sufficient to prove that R satisfies (C1') and (C2').

In order to prove that R satisfies (C1'), suppose $\rho_1 u \in SFS$ and $t_1 \in u$. Then for all $t'' \in u - t_1$, $(t_1, t'') \in Ind_N$. This implies that $\rho_1 t_1 \, R \, \rho_1 (u - t_1) t_1$ which proves that R satisfies (C1).

Now in order to prove that R satisfies (C2'), suppose $\rho_1 t_1, \rho_1' t_1 \in PI$ are such that $\rho_1 \hat{\simeq} \rho_1'$. Then for all $t_2 \in T$, $num_{t_2}(\rho_1) = num_{t_2}(\rho_1')$ and hence $\rho_1 t_1 \, R \, \rho_1' t_1$. This proves that R also satisfies (C2'). □

Note that in the above lemma we have in particular that prime intervals associated with a transition which are equivalent under $\hat{\sim}$, have the same number of occurrences of this transition. This property, which is similar to Lemma 5.3.4(1), follows from the irreflexivity of Ind_N.

In Lemma 5.6.2 it has been proved that M-equivalent firing sequences of a 1-safe Petri net N have the same past (under $\approx_N$). Now we show that we have a similar property for the equivalence relation $\hat{\simeq}_N$.

Lemma 5.6.6

Let N be a 1-safe Petri net and let $\rho, \rho' \in SFS$ be such that $\rho \;\hat{\doteq}\; \rho'$. Then $past_{\sim}(\rho) = past_{\sim}(\rho')$.

Proof.

Suppose $\rho = \rho_1 u v \rho_2$, $\rho' = \rho_1 u' v' \rho_2$, $u \cup v = u' \cup v'$, $u \cap v = u' \cap v' = \emptyset$, and, for all $t, t' \in u \cup v$ with $t \neq t'$, $(t, t') \in Ind_N$. Then by Lemma 2.1.10, $\rho_1(u \cup v) \in SFS$. From (C1') it now easily follows that, for all $t \in v$, $\rho_1 u t \hat{\sim} \rho_1 t$ and, for all $t \in v'$, $\rho_1 u' t \hat{\sim} \rho_1 t$. Hence $past_{\sim}(\rho_1 u v) = past_{\sim}(\rho_1 u' v')$. Since $\rho_1 u v \sigma \;\hat{\doteq}\; \rho_1 u' v' \sigma$ for every prefix σ of ρ_2, it easily follows from (C2') that also $past_{\sim}(\rho) = past_{\sim}(\rho')$. □

Now we can prove the inclusion $\approx_N \subseteq \hat{\sim}_N$.

Lemma 5.6.7

Let $N = (S, T, W, M_{in})$ be a 1-safe Petri net and let $\rho t, \rho' t' \in PI$ be such that $\rho t \approx \rho' t'$. Then $\rho t \,\hat{\sim}\, \rho' t'$.

Proof.

It is sufficient to prove that $\hat{\sim}$ satisfies (C1) and (C2) because then $\approx \subseteq \hat{\sim}$.

We immediately have that $\hat{\sim}$ satisfies (C1) because $\hat{\sim}$ satisfies (C1'). In order to prove that $\hat{\sim}$ satisfies (C2), suppose $\rho_1 t_1, \rho_1' t_1 \in PI$ are such that $past_{\sim}(\rho_1) = past_{\sim}(\rho_1')$. Let $\rho_2, \rho_2' \in FS$ be such that $\rho_1 \hat{\simeq} \rho_2$ and $\rho_1' \hat{\simeq} \rho_2'$. It is now sufficient to prove that $\rho_2 \hat{\simeq} \rho_2'$ because then $\rho_1 \hat{\simeq} \rho_2 \hat{\simeq} \rho_2' \hat{\simeq} \rho_1'$. This would imply that $\rho_1 t_1 \,\hat{\sim}\, \rho_1' t_1$ because $\hat{\sim}$ satisfies (C2').

In order to prove that $\rho_2 \hat{\simeq} \rho_2'$, first note that repeatedly applying Lemma 5.6.6 implies, $past_{\sim}(\rho_2) = past_{\sim}(\rho_1) = past_{\sim}(\rho_1') = past_{\sim}(\rho_2')$. We now proceed by induction on $k = |\rho_2|$.

If $k = 0$ then $\rho_2 \hat{\simeq} \rho_2'$ clearly holds, so assume that $k > 0$. Let $\rho_2 = \rho_3 t_3$. Then because $past_{\sim}(\rho_2) = past_{\sim}(\rho_2')$ there exist $\rho_3', \rho_3'' \in T^*$ such that $\rho_2' = \rho_3' t_3 \rho_3''$ and $\rho_3 t_3 \,\hat{\sim}\, \rho_3' t_3$. Moreover, from Lemma 5.6.5 it easily follows that for all $t_2 \in T$, $num_{t_2}(\rho_2) = num_{t_2}(\rho_2')$. Hence we must have by Lemma 5.6.5 that for all $t_2 \in alph(\rho_3'')$, $(t_2, t_3) \in Ind_N$. This now implies that $\rho_2' \hat{\simeq} \rho_3' \rho_3'' t_3$. Now by Lemma 5.6.6, $past_{\sim}(\rho_2) = past_{\sim}(\rho_2') = past_{\sim}(\rho_3' \rho_3'' t_3)$. Then by Lemma 5.6.5 we also have that $past_{\sim}(\rho_3) = past_{\sim}(\rho_3' \rho_3'')$. Then $\rho_3 \hat{\simeq} \rho_3' \rho_3''$ by the induction hypothesis. Hence also $\rho_2 = \rho_3 t_3 \hat{\simeq} \rho_3' \rho_3'' t_3 \hat{\simeq} \rho_2'$. This proves that $\hat{\sim}$ satisfies (C2). □

Thus (sequential) prime intervals which are equivalent under $\approx_N$ are also equivalent under $\hat{\sim}_N$. Next we show that for sequential prime intervals this further extends to equivalence under $\sim_N$.

Lemma 5.6.8

Let N be a 1-safe Petri net and let $\rho t, \rho' t' \in sPI$ be such that $\rho t \,\hat{\sim}\, \rho' t'$. Then $\rho t \sim \rho' t'$.

Proof.

In order to prove that $\rho t \sim \rho' t'$, define the relation $R \subseteq PI \times PI$ by: $\rho_1 t_1 \, R \, \rho_2 t_2$ iff $t_1 = t_2$ and $\exists \rho_1' t_1, \rho_2' t_2 \in sPI.\, (\rho_1 \hat{\simeq} \rho_1'$ and $\rho_2 \hat{\simeq} \rho_2'$ and $\rho_1' t_1 \sim \rho_2' t_2)$. Assume for the moment that R is an equivalence relation which satisfies (C1') and (C2'). Then $\hat{\sim} \subseteq R$ because $\hat{\sim}$ is the least equivalence relation satisfying (C1') and (C2'). This implies that $\rho t \, R \, \rho' t'$, so there exist $\rho_1 t, \rho_1' t' \in sPI$ such that $\rho \hat{\simeq} \rho_1$, $\rho' \hat{\simeq} \rho_1'$, and $\rho_1 t \sim \rho_1' t'$. Then by Lemma 4.6.1, $\rho \simeq \rho_1$ and $\rho' \simeq \rho_1'$. Now $\rho t \sim \rho_1 t$ and $\rho' t' \sim \rho_1' t'$ because $\sim$ satisfies (S2) and thus $\rho t \sim \rho' t'$ by the transitivity of $\sim$. So it is sufficient to prove that R is an equivalence relation which satisfies (C1') and (C2').

It is easy to see that R is reflexive and symmetric. In order to prove transitivity, suppose $\rho_1 t_1 \, R \, \rho_2 t_2 \, R \, \rho_3 t_3$. From $\rho_1 t_1 \, R \, \rho_2 t_2$ we know that there exist $\rho_1' t_1, \rho_2' t_2 \in sPI$ such that $\rho_1 \hat{\simeq} \rho_1'$, $\rho_2 \hat{\simeq} \rho_2'$, and $\rho_1' t_1 \sim \rho_2' t_2$. Similarly, since $\rho_2 t_2 \, R \, \rho_3 t_3$, there exist $\rho_2'' t_2, \rho_3' t_3 \in sPI$ such that $\rho_2 \hat{\simeq} \rho_2''$, $\rho_3 \hat{\simeq} \rho_3'$, and $\rho_2'' t_2 \sim \rho_3' t_3$. Then $\rho_2' \hat{\simeq} \rho_2''$ and hence

also $\rho_2' \simeq \rho_2''$ by Lemma 4.6.1. Hence $\rho_2' t_2 \sim \rho_2'' t_2$ because $\sim$ satisfies (S2). This implies that $\rho_1' t_1 \sim \rho_3' t_3$ by the transitivity of $\sim$. We can now conclude that $\rho_1 t_1 \, R \, \rho_3 t_3$. This proves that R is an equivalence relation.

In order to prove that R satisfies (C1'), suppose $\rho_1 u \in SFS$ and $t_1 \in u$. It must be proved that $\rho_1 t_1 \, R \, \rho_1(u - t_1)t_1$. We proceed by induction on $k = |u|$. If $k = 1$ then the claim holds by the reflexivity of R. Now assume that $k > 1$. Then let $t_2 \in u - t_1$. By the induction hypothesis, $\rho_1 t_1 \, R \, \rho_1(u - \{t_1, t_2\})t_1$. By the transitivity of R it is then sufficient to prove that $\rho_1(u - \{t_1, t_2\})t_1 \, R \, \rho_1(u - t_1)t_1$. Let $\rho_1' \in FS$ be such that $\rho_1(u - \{t_1, t_2\}) \hat{\simeq} \rho_1'$. Then clearly $\rho_1(u - t_1) \hat{\simeq} \rho_1(u - \{t_1, t_2\})t_2 \hat{\simeq} \rho_1' t_2$. Because $\sim$ satisfies (S1) and $(t_1, t_2) \in Ind_N$, we also have that $\rho_1' t_1 \sim \rho_1' t_2 t_1$. Then by the definition of R, $\rho_1(u - \{t_1, t_2\})t_1 \, R \, \rho_1(u - t_1)t_1$. This proves that R satisfies (C1').

Now in order to prove that R satisfies (C2'), suppose $\rho_1 t_1, \rho_1' t_1 \in PI$ are such that $\rho_1 \hat{\simeq} \rho_1'$. Let $\rho_2, \rho_2' \in FS$ be such that $\rho_1 \hat{\simeq} \rho_2$ and $\rho_1' \hat{\simeq} \rho_2'$. Hence also $\rho_2 \hat{\simeq} \rho_2'$ by the transitivity of $\hat{\simeq}$. By Lemma 4.6.1 we then have that $\rho_2 \simeq \rho_2'$. Thus $\rho_2 t_1 \sim \rho_2' t_1$ bacause $\sim$ satisfies (S2) and hence $\rho_1 t_1 \, R \, \rho_1' t_1$ by the definition of R. This proves that R satisfies (C2'). We can now conclude that $\rho t \sim \rho' t'$. □

Now we can prove the result we are after.

Theorem 5.6.9

Let $N = (S, T, W, M_{in})$ be a 1-safe Petri net. Then $nu(N) \equiv pu(sp(N))$.

Proof.

Let $nu(N) = (E, C, \vdash)$, $sp(N) = (\hat{E}, \leq, \#)$, and $pu(sp(N)) = (\hat{E}, FC_{sp(N)}, \hat{\vdash})$. We prove that $\lambda_N : \hat{E} \to E$ given by $\lambda_N(\langle \rho t \rangle_{\sim_N}) = \langle \rho t \rangle_N$ is an LES-isomorphism from $pu(sp(N))$ to $nu(N)$. By Lemma 5.6.3, λ_N is well-defined, by Lemma 5.6.4 λ_N is surjective, and by Lemma 5.6.7 and Lemma 5.6.8 λ_N is injective. Moreover, $C = \{past_N(\rho) \mid \rho \in SFS\} = \{past_N(\rho) \mid \rho \in FS\} = \{\lambda_N(ev_N(\rho)) \mid \rho \in FS\}$. Hence $C = \lambda_N(FC_{sp(N)})$ by Lemma 5.1.6. Now let $c \in FC_{sp(N)}$ and $u \in P_F(\hat{E})$. Then it must be proved that $c \, \hat{\vdash} \, u$ iff $\lambda_N(c) \vdash \lambda_N(u)$.

First assume that $\lambda_N(c) \vdash \lambda_N(u)$. Then by Lemma 5.4.2, $\lambda_N(c) \cap \lambda_N(u) = \emptyset$ and, for all $v \subseteq \lambda_N(u)$, $\lambda_N(c) \cup v \in C = \lambda_N(FC_{sp(N)})$. Since λ_N is a bijection, this implies that also $c \cap u = \emptyset$ and, for all $v \subseteq u$, $c \cup v \in FC_{sp(N)}$. Then by the definition of $\hat{\vdash}$, $c \, \hat{\vdash} \, u$.

Now assume that $c \, \hat{\vdash} \, u$. It must be proved that $\lambda_N(c) \vdash \lambda_N(u)$. By the definition of $\hat{\vdash}$, $c \cap u = \emptyset$ and, for all $v \subseteq u$, $c \cup v \in FC_{sp(N)}$. This implies that for all $\langle \rho_1 t_1 \rangle_{\sim_N}, \langle \rho_2 t_2 \rangle_{\sim_N} \in u$ with $\langle \rho_1 t_1 \rangle_{\sim_N} \neq \langle \rho_2 t_2 \rangle_{\sim_N}$, $\langle \rho_1 t_1 \rangle_{\sim_N} \, co_{sp(N)} \, \langle \rho_2 t_2 \rangle_{\sim_N}$, and hence by Lemma 5.1.7, $(t_1, t_2) \in Ind_N$. By Lemma 5.1.6 there exists $\rho \in FS$ such that $ev_N(\rho) = c$. We now prove that, for all $\langle \rho' t \rangle_{\sim_N} \in u$, $\rho t \in FS$ and $\langle \rho t \rangle_{\sim_N} = \langle \rho' t \rangle_{\sim_N}$. Suppose $\langle \rho' t \rangle_{\sim_N} \in u$. Then $c \cup \langle \rho' t \rangle_{\sim_N} \in FC_{sp(N)}$, so there exists $\rho'' \in FS$ with $ev_N(\rho'') = c \cup \langle \rho' t \rangle_{\sim_N}$. Moreover, $[\rho]_{Ind_N} \preceq [\rho'']_{Ind_N}$ by Lemma 5.1.3. From Lemma 5.1.2 it easily follows that $mset(\rho'') = mset(\rho) + t$. Hence we must have that $\rho t \simeq \rho''$. This implies that $\rho t \in FS$ and, by Lemma 5.1.3, $c \cup \langle \rho t \rangle_{\sim_N} = c \cup \langle \rho' t \rangle_{\sim_N}$. Hence by Lemma 5.1.2, $\langle \rho t \rangle_{\sim_N} = \langle \rho' t \rangle_{\sim_N}$. From Lemma 2.1.10 it now easily follows that $\rho v \in SFS$ where $v = \{t \in T \mid \langle \rho' t \rangle_{\sim_N} \in u\}$. Now by the definition of $\vdash$,

$\lambda_N(c) = past_N(\rho) \vdash \{\langle \rho t \rangle_N \mid t \in v\} = \lambda_N(u)$. This proves that λ_N is an LES-isomorphism from $pu(sp(N))$ to $nu(N)$. □

Thus our event structure semantics for Petri nets, when restricted to 1-safe Petri nets, agrees completely (up to isomorphism) with the event structure semantics of [66] for 1-safe Petri nets. Hence Theorem 5.4.7 and Theorem 5.6.9 together assure us that our event structure semantics for Petri nets (even with auto-concurrency filtered out) is an extension of the basic result in [66]. Moreover, the UL-event structure ES_3 from Example 5.4.3 and Example 5.2.2 is an example of a PN-event structure which is not the image under pu of any prime event structure, so that the extension is *strict*. This contrasts with the event structure semantics for Petri nets given in [60] where the same class of (prime) event structures is used for representing the behaviour of both 1-safe and general Petri nets.

5.7 A Co-reflection Between $\mathcal{ULES}$ and $\mathcal{PNS}$

The back-and-forth constructions established in [66] between 1-safe Petri nets and prime event structures were later proved by Winskel [96] to be the "right" ones. He achieved this by equipping both classes of objects with suitable behaviour-preserving morphisms and showed that the constructions of [66] smoothly lift to a pair of functors which constitute a co-reflection. Our aim here is to explore to what extent we can mimic this categorical result in the present, much richer setting. We show that due to auto-concurrency we cannot obtain a co-reflection between the categories of UL-event structures and Petri nets. We do however get a co-reflection for the subcategory of co-safe Petri nets. This is the main result of this section. A consequence of this result is that the category of UL-event structures is a full co-reflective subcategory of the category of L-event structures.

The notion of LES-morphism defined in Section 5.2 leads to the following definition.

Definition 5.7.1

Let $\mathcal{LES}$ be the category which has L-event structures as its objects and LES-morphisms as its arrows. The identity morphism associated with an object ES is id_{ES}; composition of LES-morphisms is composition of partial functions.

Let $\mathcal{ULES}$ be the full subcategory of $\mathcal{LES}$ the objects of which are UL-event structures. □

We are looking for a co-reflection between $\mathcal{ULES}$ and $\mathcal{PN}$ in which the left adjoint would act as en on the objects of $\mathcal{ULES}$ and the right adjoint would act as nu on the objects of $\mathcal{PN}$.

To achieve this, we would like to extend the map nu to become a functor from $\mathcal{PN}$ to $\mathcal{ULES}$ in such a way that prime intervals are preserved. This means that whenever (β, η) is a PN-morphism from N to N' and $\langle \rho t \rangle_N$ is an event of $nu(N)$ such that $\eta(t)$ is defined, then $nu((\beta, \eta))(\langle \rho t \rangle_N)$ is defined. Unfortunately, as the next example shows, this is not possible.

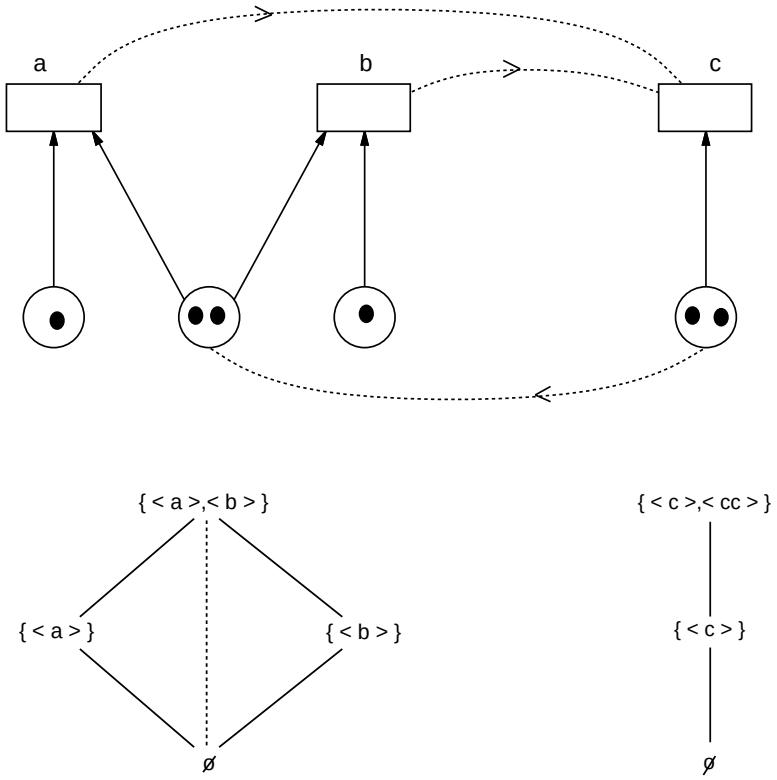


Figure 5.8: A PN-morphism from N to N' with the UL-event structures $nu(N)$ and $nu(N')$

Example 5.7.2

Let (β, η) be the PN-morphism from N to N' from Figure 2.7. This PN-morphism is depicted together with the UL-event structures $nu(N)$ and $nu(N')$ in Figure 5.8. The UL-event structure $nu(N)$ has two events, $\langle a \rangle_N = \langle ba \rangle_N$ and $\langle b \rangle_N = \langle ab \rangle_N$. Also the UL-event structure $nu(N')$ has two events, $\langle c \rangle_{N'}$ and $\langle cc \rangle_{N'}$. Even though both $\eta(a)$ and $\eta(b)$ are defined, there exists however no LES-morphism f from $nu(N)$ to $nu(N')$ in which both $f(\langle a \rangle_N)$ and $f(\langle b \rangle_N)$ are defined. Thus we cannot extend the map nu to a functor in this way. □

The problem is that in a PN-morphism transitions which can occur concurrently, may be mapped to the same transition, leading to auto-concurrency. As a consequence, *step* firing sequences of the first Petri net may be mapped to *multiset* firing sequences of the second Petri net. For this reason we restrict our attention to co-safe Petri nets in the rest of this section.

In what follows the map nu defined in Section 5.4, when restricted to co-safe Petri nets, is extended to a functor from $\mathcal{PNS}$, the category of co-safe Petri nets defined in Section 2.2, to $\mathcal{ULES}$. Then the map en defined in Section 5.5 is extended to a functor from $\mathcal{LES}$ to $\mathcal{PNS}$. Once these functors are defined we can prove the desired co-reflection between $\mathcal{ULES}$ and $\mathcal{PNS}$.

From Lemma 2.2.2 we already know that for co-safe Petri nets prime intervals are preserved under PN-morphisms. In the following lemma it is proved that for co-safe Petri nets also equivalence of prime intervals is preserved under PN-morphisms.

Lemma 5.7.3

Let $N_i = (S_i, T_i, W_i, M_i)$, $i = 1, 2$, be co-safe Petri nets and let (β, η) be a PN-morphism from N_1 to N_2. Let $t \in T$ be such that $\eta(t)$ is defined and let $\rho t, \rho' t \in PI_{N_1}$ be such that $\rho t \approx_{N_1} \rho' t$. Then $\eta(\rho)\eta(t) \approx_{N_2} \eta(\rho')\eta(t)$.

Proof.

Define $R \subseteq PI_{N_1} \times PI_{N_1}$ by: $\rho_1 t_1 \, R \, \rho_2 t_2$ iff ($t_1 = t_2$ and $\eta(t_1)$ is undefined) or ($\eta(t_1)$ and $\eta(t_2)$ are defined and $\eta(\rho_1)\eta(t_1) \approx_{N_2} \eta(\rho_2)\eta(t_2)$). Note that R is an equivalence relation. Suppose R is SFS_{N_1}-consistent. Then since $\approx_{N_1}$ is the least equivalence relation which is SFS_{N_1}-consistent, it follows that $\approx_{N_1} \subseteq R$. Hence $\rho t \, R \, \rho' t$ and thus, by the definition of R, $\eta(\rho)\eta(t) \approx_{N_2} \eta(\rho')\eta(t)$. Thus it is sufficient to prove that R satisfies the conditions (C1) and (C2).

Suppose $\rho_1 u \in SFS_{N_1}$ and $t_1 \in u$. If $\eta(t_1)$ is undefined then we immediately have that $\rho_1 t_1 R \rho_1 (u - t_1) t_1$, so assume that $\eta(t_1)$ is defined. Then $\eta(\rho_1 u) \in SFS_{N_2}$ by Lemma 2.2.2 and $\eta(t_1) \in \eta(u)$. Since $\approx_{N_2}$ satisfies (C1), it then follows that $\eta(\rho_1)\eta(t_1) \approx_{N_2} \eta(\rho_1)(\eta(u) - \eta(t_1))\eta(t_1)$. Moreover, by Lemma 2.2.2 and the fact that N_2 is co-safe we have that $\eta(\rho_1)(\eta(u) - \eta(t_1)) = \eta(\rho_1(u - t_1))$. This yields $\rho_1 t_1 \, R \, \rho_1 (u - t_1) t_1$ by the definition of R. Thus R satisfies (C1).

Now suppose $\sigma t', \sigma' t' \in PI_{N_1}$ are such that $past_R(\sigma) = past_R(\sigma')$. If $\eta(t')$ is undefined then we immediately have that $\sigma t' \, R \, \sigma' t'$, so assume that $\eta(t')$ is defined. Suppose $past_{N_2}(\eta(\sigma)) = past_{N_2}(\eta(\sigma'))$. Then since $\approx_{N_2}$ satisfies (C2) we know that $\eta(\sigma)\eta(t') \approx_{N_2} \eta(\sigma')\eta(t')$ and hence $\sigma t' \, R \, \sigma' t'$. Thus in order to prove that R satisfies (C2), it is sufficient to prove that $past_{N_2}(\eta(\sigma)) = past_{N_2}(\eta(\sigma'))$.

Let $\langle \rho_1 t_1 \rangle_{N_2} \in past_{N_2}(\eta(\sigma))$. Then there exists $\rho_2 t_2 \in int(\sigma)$ such that $\eta(t_2)$ is defined and $\langle \rho_1 t_1 \rangle_{N_2} = \langle \eta(\rho_2)\eta(t_2) \rangle_{N_2}$. Then also $\langle \rho_2 t_2 \rangle_R \in past_R(\sigma) = past_R(\sigma')$. Hence there exists $\rho_3 t_3 \in int(\sigma')$ such that $\langle \rho_2 t_2 \rangle_R = \langle \rho_3 t_3 \rangle_R$. Since $\eta(t_2)$ is defined this implies that $\eta(t_3)$ is also defined and $\langle \eta(\rho_2)\eta(t_2) \rangle_{N_2} = \langle \eta(\rho_3)\eta(t_3) \rangle_{N_2}$. Moreover, $\langle \eta(\rho_3)\eta(t_3) \rangle_{N_2} \in past_{N_2}(\eta(\sigma'))$ by the definition of *past*. Hence $\langle \rho_1 t_1 \rangle_{N_2} \in past_{N_2}(\eta(\sigma'))$. This proves that $past_{N_2}(\eta(\sigma)) \subseteq past_{N_2}(\eta(\sigma'))$. Similarly we have $past_{N_2}(\eta(\sigma')) \subseteq past_{N_2}(\eta(\sigma))$ and thus $past_{N_2}(\eta(\sigma)) = past_{N_2}(\eta(\sigma'))$. □

Now we can extend the map nu to a functor, also denoted by nu, from $\mathcal{PNS}$ to $\mathcal{ULES}$.

Let N_1 and N_2 be a pair of co-safe Petri nets and let (β, η) be a PN-morphism from N_1 to N_2. Suppose $nu(N_1) = (E_1, C_1, \vdash_1)$ and $nu(N_2) = (E_2, C_2, \vdash_2)$. Then we define $nu((\beta, \eta))$ to be the partial function from E_1 to E_2 given by:

$$\forall \langle \rho t \rangle_{N_1} \in E_1 . \, nu((\beta, \eta))(\langle \rho t \rangle_{N_1}) = \begin{cases} \text{undefined} & \text{if } \eta(t) \text{ is undefined} \\ \langle \eta(\rho)\eta(t) \rangle_{N_2} & \text{otherwise.} \end{cases}$$

Note that by Lemma 5.7.3, $nu((\beta, \eta))$ is well-defined.

Lemma 5.7.4

Let N_1 and N_2 be co-safe Petri nets and let (β, η) be a PN-morphism from N_1 to N_2. Then $nu((\beta, \eta))$ is an LES-morphism from $nu(N_1)$ to $nu(N_2)$.

Proof.

Let $nu(N_1) = (E_1, C_1, \vdash_1)$ and $nu(N_2) = (E_2, C_2, \vdash_2)$. Let $nu((\beta, \eta))$ be denoted by f. Given $\hat{c} \vdash_1 \hat{u}$ we have to prove that $f(\hat{c}) \vdash_2 f(\hat{u})$. So suppose $\hat{c} \vdash_1 \hat{u}$. Then there exists $\rho u \in SFS_{N_1}$ such that $\hat{c} = past_{N_1}(\rho)$ and $\hat{u} = \{\langle \rho t \rangle_{N_1} \mid t \in u\}$. By Lemma 2.2.2 we have that $\eta(\rho), \eta(\rho u) \in SFS_{N_2}$. Hence by the definition of $\vdash_2$ $past_{N_2}(\eta(\rho)) \vdash_2 \{\langle \eta(\rho) t' \rangle_{N_2} \mid t' \in \eta(u)\}$. Now $past_{N_2}(\eta(\rho)) = \{\langle \rho_2 t_2 \rangle_{N_2} \mid \rho_2 t_2 \in int(\eta(\rho))\} = \{\langle \eta(\rho_1)\eta(t_1) \rangle_{N_2} \mid \rho_1 t_1 \in int(\rho) \text{ with } \eta(t_1) \text{ defined } \} = f(past_{N_1}(\rho)) = f(\hat{c})$. Furthermore, $\{\langle \eta(\rho) t' \rangle_{N_2} \mid t' \in \eta(u)\} = \{\langle \eta(\rho)\eta(t) \rangle_{N_2} \mid t \in u \text{ with } \eta(t) \text{ defined}\} = f(\hat{u})$. And so $f(\hat{c}) \vdash_2 f(\hat{u})$ as required. □

From the definition of nu it easily follows that nu preserves identities and respects composition. Hence the following result follows from Theorem 5.4.7 and Lemma 5.7.4.

Theorem 5.7.5

nu is a functor from $\mathcal{PNS}$ to $\mathcal{ULES}$. □

Next the map en is extended to a functor - also denoted by en - from $\mathcal{LES}$ to $\mathcal{PNS}$. Then we show that this functor is in fact full and faithful. On arrows, en is defined by interpreting morphisms between L-event structures as morphisms between their associated multiset transition systems under a map et, and then applying the map tn from MTS-morphisms to PN-morphisms as defined in Section 3.2.

Let $ES_i = (E_i, C_i, \vdash_i)$, $i = 1, 2$, be a pair of L-event structures and let f be an LES-morphism from ES_1 to ES_2. Then define $et(f) = (f, g)$ where $g : C_1 \to C_2$ is given by $g(c) = f(c)$. Then it is easy to see that $et(f)$ is an MTS-morphism from $et(ES_1)$ to $et(ES_2)$. Hence by Lemma 3.2.9 there exists for every region r of ES_2 an inverse region $et(f)^{-1}(r)$ of ES_1. In what follows we denote $et(f)^{-1}(r)$ by $f^{-1}(r)$. So for a region r of ES_2, $f^{-1}(r) : C_1 \cup E_1 \to \mathbf{N} \cup (\mathbf{N} \times \mathbf{N})$ is given by:

(1) $\forall c \in C_1 . f^{-1}(r)(c) = r(f(c))$

(2) $\forall e \in E_1 . f^{-1}(r)(e) = \begin{cases} r(f(e)) & \text{if } f(e) \text{ is defined} \\ (0,0) & \text{otherwise.} \end{cases}$

Now define $en(f) = tn(et(f))$. So $en(f) = (\beta_f, \eta_f)$ where $\eta_f = f$ and $\beta_f : \mathcal{R}_{ES_2} \to \mathcal{R}_{ES_1}$ is given by:

$$\beta_f(r) = \begin{cases} f^{-1}(r) & \text{if } f^{-1}(r) \text{ is non-trivial} \\ \text{undefined} & \text{otherwise.} \end{cases}$$

From Lemma 3.2.10 we now immediately have the following result.

Lemma 5.7.6

Let ES_1 and ES_2 be L-event structures and let f be an LES-morphism from ES_1 to ES_2. Then $en(f)$ is a PN-morphism from $en(ES_1)$ to $en(ES_2)$. □

Now we are ready to prove that en is a functor, which is full and faithful.

Theorem 5.7.7
en is a full and faithful functor from $\mathcal{LES}$ to $\mathcal{PNS}$.

Proof.
In order to prove that en is a functor from $\mathcal{LES}$ to $\mathcal{PNS}$, it is by Lemma 3.2.10 and Lemma 5.7.6 sufficient to prove that en preserves identities and respects composition. Clearly en preserves identities. Assume that f_1 is an LES-morphism from ES_1 to ES_2 and f_2 is an LES-morphism from ES_2 to ES_3. We have that $\eta_{f_2 \circ f_1} = f_2 \circ f_1 = \eta_{f_2} \circ \eta_{f_1}$. Because $en(ES)$ is S-simple we have by Lemma 2.2.4 that $en(f_2 \circ f_1) = (\beta_{f_2 \circ f_1}, \eta_{f_2 \circ f_1}) = (\beta_{f_1} \circ \beta_{f_2}, \eta_{f_2} \circ \eta_{f_1}) = (\beta_{f_2}, \eta_{f_2}) \circ (\beta_{f_1}, \eta_{f_1}) = en(f_2) \circ en(f_1)$.

In order to prove that en is full, let $ES_1 = (E_1, C_1, \vdash_1)$ and $ES_2 = (E_2, C_2, \vdash_2)$ be L-event structures and let (β, η) be a PN-morphism from $en(ES_1)$ to $en(ES_2)$. We first prove that η is an LES-morphism from ES_1 to ES_2. Suppose $c \vdash_1 u$. Let $\rho \in SFS_{ES_1}$ be such that $alph(\rho) = c$. Then $\rho u \in SFS_{ES_1}$ and hence we also have, by Lemma 5.5.3, that $\rho u \in SFS_{en(ES_1)}$. By Lemma 2.2.2 we then have that $\eta(\rho u) \in SFS_{en(ES_2)}$. Again by Lemma 5.5.3 we now have that $\eta(\rho u) \in SFS_{ES_2}$. Hence $alph(\eta(\rho)) \vdash_2 \eta(u)$. Because $alph(\eta(\rho)) = \eta(c)$ we can now conclude that $\eta(c) \vdash_2 \eta(u)$. This proves that η is an LES-morphism from ES_1 to ES_2. Since $en(ES_1)$ is S-simple Lemma 2.2.4 can be applied and so $en(\eta) = (\beta, \eta)$. This proves that en is full.

Finally, if f and g are LES-morphisms from ES_1 to ES_2 such that $f \neq g$ then also $en(f) \neq en(g)$ by the definition of en. Hence en is faithful. □

Next we show that $en \circ i$ and nu form a co-reflection with $en \circ i$ as the left adjoint, where i is the inclusion functor from $\mathcal{ULES}$ to $\mathcal{LES}$.

In order to facilitate the proof of this result we first define the PN-morphisms which turn out to form the co-unit of the adjunction. To do this the following regions of the L-event structure associated with a co-safe Petri net are defined.

Let $N = (S, T, W, M_{in})$ be a co-safe Petri net with $nu(N) = (E, C, \vdash)$ and let $s \in S$. Define $r_s : C \cup E \to \mathbf{N} \cup (\mathbf{N} \times \mathbf{N})$ by:

(1) $\forall \rho \in SFS_N . r_s(past_N(\rho)) = M_\rho(s)$

(2) $\forall \langle \rho t \rangle_N \in E . r_s(\langle \rho t \rangle_N) = (W(s,t), W(t,s))$.

From Lemma 5.3.4(1) it easily follows that part (1) in the definition of r_s is well-defined.

Lemma 5.7.8
Let $N = (S, T, W, M_{in})$ be a co-safe Petri net and let $s \in S$. Then r_s is a region of $nu(N)$.

Proof.
Let $nu(N) = (E, C, \vdash)$. Suppose $\hat{c} \vdash \hat{u}$. Then there is $\rho u \in SFS_N$ such that $\hat{c} = past_N(\rho)$ and $\hat{u} = \{\langle \rho t \rangle_N \mid t \in u\}$. Then $r_s(\hat{c}) = M_\rho(s) \geq \sum_{t \in u} W(s,t) = \sum_{t \in u} {}^{r_s}\langle \rho t \rangle_N$ and $r_s(\hat{c} \cup \hat{u}) = M_{\rho u}(s) = M_\rho(s) + \sum_{t \in u}(W(t,s) - W(s,t)) = r_s(\hat{c}) + \sum_{t \in u}(\langle \rho t \rangle_N{}^{r_s} - {}^{r_s}\langle \rho t \rangle_N)$. □

For a co-safe Petri net $N = (S, T, W, M_{in})$ with $nu(N) = (E, C, \vdash)$ and $en(nu(N)) = (\mathcal{R}_{nu(N)}, E, \hat{W}, \hat{M}_{in})$, we define $fold_S : S \to \mathcal{R}_{nu(N)}$ and $fold_T : E \to T$ by:

(1) $\forall s \in S. fold_S(s) = \begin{cases} r_s & \text{if } r_s \text{ is non-trivial} \\ \text{undefined} & \text{otherwise.} \end{cases}$

(2) $\forall \langle \rho t \rangle_N \in E. fold_T(\langle \rho t \rangle_N) = t.$

Lemma 5.7.9

Let $N = (S, T, W, M_{in})$ be a co-safe Petri net with $nu(N) = (E, C, \vdash)$ and with $en(nu(N)) = (\mathcal{R}_{nu(N)}, E, \hat{W}, \hat{M}_{in})$. Then $(fold_S, fold_T)$ is a PN-morphism from $en(nu(N))$ to N.

Proof.

Suppose $s \in S$ is such that $fold_S(s)$ is defined. Then $\hat{M}_{in}(fold_S(s)) = \hat{M}_{in}(r_s) = r_s(\emptyset) = M_{in}(s)$ which proves condition (1) in the definition of PN-morphism.

Because $fold_T$ is a total function, condition (2) in the definition of PN-morphism trivially holds.

In order to prove condition (3), suppose $\langle \rho t \rangle_N \in E$. If $s \in fold_S^{-1}({}^\bullet\langle \rho t \rangle_N)$ then we must have that $r_s \in {}^\bullet\langle \rho t \rangle_N$, that is $\hat{W}(r_s, \langle \rho t \rangle_N) > 0$. This implies that ${}^{r_s}\langle \rho t \rangle_N > 0$ and hence $W(s, t) > 0$. This proves that $s \in {}^\bullet t = {}^\bullet fold_T(\langle \rho t \rangle_N)$. On the other hand, if $s \in {}^\bullet fold_T(\langle \rho t \rangle_N) = {}^\bullet t$, then ${}^{r_s}\langle \rho t \rangle_N = W(s, t) > 0$. Thus r_s is non-trivial and $\hat{W}(r_s, \langle \rho t \rangle_N) = {}^{r_s}\langle \rho t \rangle_N > 0$. Then $r_s \in {}^\bullet\langle \rho t \rangle_N$ and hence $s \in fold_S^{-1}({}^\bullet\langle \rho t \rangle_N)$. Moreover, $W(s, fold_T(\langle \rho t \rangle_N)) = W(s, t) = \hat{W}(r_s, \langle \rho t \rangle_N) = \hat{W}(fold_S(s), \langle \rho t \rangle_N)$. Similarly it can be proved that $fold_S^{-1}(\langle \rho t \rangle_N{}^\bullet) = fold_T(\langle \rho t \rangle_N)^\bullet$ and $W(fold_T(\langle \rho t \rangle_N), s) = \hat{W}(\langle \rho t \rangle_N, fold_S(s))$. This proves condition (3) in the definition of PN-morphism. □

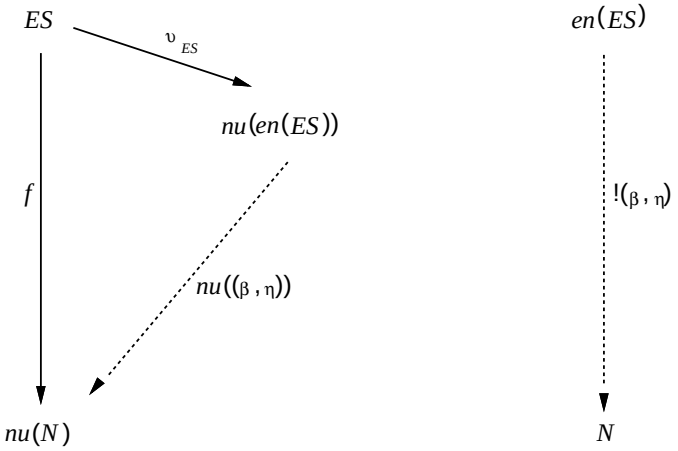
Now we can prove the main result of this section.

Theorem 5.7.10

$en \circ i : \mathcal{ULES} \to \mathcal{PNS}$ and $nu : \mathcal{PNS} \to \mathcal{ULES}$ form a co-reflection with en the left adjoint and the arrows v_{ES} as unit.

Proof.

Let $ES = (E, C, \vdash)$ be an UL-event structure, let $N = (S, T, W, M_{in})$ be a co-safe Petri net, and let f be an LES-morphism from ES to $nu(N) = (\hat{E}, \hat{C}, \hat{\vdash})$. We must show that there is a unique PN-morphism (β, η) from $en(ES) = (\mathcal{R}_{ES}, E, W_{ES}, M)$ to N such that the following diagram commutes.



Define (β, η) by $(\beta, \eta) = (fold_S, fold_T) \circ en(f)$. Hence $\beta : S \to \mathcal{R}_{ES}$ is such that for all $s \in S$, $\beta(s) = f^{-1}(r_s)$ if $f^{-1}(r_s)$ is non-trivial and $\beta(s)$ is undefined otherwise. The function $\eta : E \to T$ is such that for all $e \in E$, $\eta(e) =$ undefined if $f(e)$ is undefined and $\eta(e) = t$ if $f(e)$ is defined with $f(e) = \langle \rho t \rangle_N$. Because $(fold_S, fold_T)$ and $en(f)$ are PN-morphisms by Lemma 5.7.9 and Lemma 5.7.6 respectively, and because the composition of PN-morphisms is again a PN-morphism, the pair (β, η) is a PN-morphism.

The next thing to prove is that $nu((\beta, \eta)) \circ \upsilon_{ES} = f$. Let $e \in E$. Then $f(e)$ is undefined iff $\eta(e)$ is undefined iff $(nu((\beta, \eta)) \circ \upsilon_{ES})(e)$ is undefined. So assume that $f(e)$ is defined. Let $\rho \in SFS_{ES}$ be such that $\rho e \in SFS_{ES}$. By the unique occurrence property ρ exists. By Lemma 5.5.3 we then have that also $\rho, \rho e \in SFS_{en(ES)}$ and hence Lemma 2.2.2 implies that $\eta(\rho), \eta(\rho e) \in SFS_N$. Furthermore, by Lemma 5.2.8, $f(\rho), f(\rho e) \in SFS_{nu(N)}$.

We first prove, by induction on $|\rho|$, that $alph(f(\rho)) = past_N(\eta(\rho))$. If $\rho = \emptyset$ then this is clear, so assume that $\rho = \rho' u$ with $\rho' \in SFS_{ES}$ and $\emptyset \neq u \in P_F(E)$.

Then $alph(f(\rho)) = alph(f(\rho')) \cup f(u)$ and $past_N(\eta(\rho)) = past_N(\eta(\rho')) \cup \hat{u}$ where $\hat{u} = \{\langle \eta(\rho')\eta(e') \rangle_N \mid e' \in u \text{ with } \eta(e') \text{ defined } \}$. By the induction hypothesis, $alph(f(\rho')) = past_N(\eta(\rho'))$. From $f(\rho' u) \in SFS_{nu(N)}$ we have that $alph(f(\rho'))\ \hat{\vdash}\ f(u)$. On the other hand, from $\eta(\rho' u) \in SFS_N$ we have that $past_N(\eta(\rho'))\ \hat{\vdash}\ \hat{u}$. It is now sufficient to prove that $f(u) = \hat{u}$. By the definition of $\hat{\vdash}$, $alph(f(\rho'))\ \hat{\vdash}\ f(u)$ implies that there exists $\rho_1 u_1 \in SFS_N$ such that $alph(f(\rho')) = past_N(\rho_1)$ and $f(u) = \{\langle \rho_1 e_1 \rangle_N \mid e_1 \in u_1\}$. Let $e' \in u$ be such that $f(e')$ is defined. Then there exists $e_1 \in u_1$ such that $f(e') = \langle \rho_1 e_1 \rangle_N$. Then $e_1 = \eta(e')$ by the definition of η. Since $past_N(\rho_1) = alph(f(\rho')) = past_N(\eta(\rho'))$ and $\approx_N$ satisfies (C2), we must now have that $\langle \eta(\rho')\eta(e') \rangle_N = \langle \rho_1 e_1 \rangle_N$. This proves that $f(u) = \hat{u}$ and we can conclude that $alph(f(\rho)) = past_N(\eta(\rho))$.

From $f(\rho e) \in SFS_{nu(N)}$ we know that $alph(f(\rho))\ \hat{\vdash}\ f(e)$. Then there exists $\rho_2 e_2 \in SFS_N$ such that $alph(f(\rho)) = past_N(\rho_2)$ and $f(e) = \langle \rho_2 e_2 \rangle_N$. Then $e_2 = \eta(e)$ by the definition of η. Since $past_N(\rho_2) = alph(f(\rho)) = past_N(\eta(\rho))$ and $\approx_N$ satisfies (C2), we now have that $\langle \rho_2 e_2 \rangle_N = \langle \eta(\rho)\eta(e) \rangle_N$. This implies that $(nu((\beta, \eta)) \circ \upsilon_{ES})(e) = nu((\beta, \eta))(\langle \rho e \rangle_{en(ES)}) = \langle \eta(\rho)\eta(e) \rangle_N = \langle \rho_2 e_2 \rangle_N = f(e)$ what had to be proved.

Finally, in order to prove that (β, η) is the unique PN-morphism from $en(ES)$ to N such that $nu((\beta, \eta)) \circ \upsilon_{ES} = f$, assume that (β', η') is any PN-morphism from $en(ES)$

to N such that $nu((\beta', \eta')) \circ \upsilon_{ES} = f$. Then for all $e \in E$, $\eta(e)$ is undefined iff $f(e)$ is undefined iff $\eta'(e)$ is undefined. Now let $e \in E$ be such that $\eta'(e)$ is defined. Let $\rho \in SFS_{en(ES)}$ be such that $\upsilon_{ES}(e) = \langle \rho e \rangle_{en(ES)}$.

Then $\langle \eta(\rho)\eta(e) \rangle_N = nu((\beta, \eta)) \circ \upsilon_{ES}(e) = f(e) = nu((\beta', \eta')) \circ \upsilon_{ES}(e) = \langle \eta'(\rho)\eta'(e) \rangle_N$. Now Lemma 5.3.4(1) guarantees that $\eta(e) = \eta'(e)$. This proves that $\eta = \eta'$. We can now conclude by Lemma 2.2.4 that $\beta = \beta'$ because $en(ES)$ is S-simple.

This proves that $en \circ i$ and nu form an adjunction with $en \circ i$ as the left adjoint and the arrows υ_{ES} as unit. By Lemma 5.5.4 the arrows υ_{ES} are LES-isomorphisms and so the adjunction is even a co-reflection. □

It is easy to verify that the arrows $(fold_S, fold_T)$ form the co-unit of the adjunction between $\mathcal{ULES}$ and $\mathcal{PNS}$.

Each UL-event structure ES is by Lemma 5.5.4 isomorphic to the UL-event structure $nu(en(ES))$. Hence for each co-safe Petri net N, $en(nu(N))$ yields an UL-event structure which is isomorphic to the UL-event structure yielded by N. The Petri net $en(nu(N))$ has a number of other interesting properties. It is saturated with respect to the places and each transition can occur exactly once. Hence the Petri net $en(nu(N))$ may be viewed as a "behavioural unfolding" of N. The associated "fold morphism" is $(fold_S, fold_T)$.

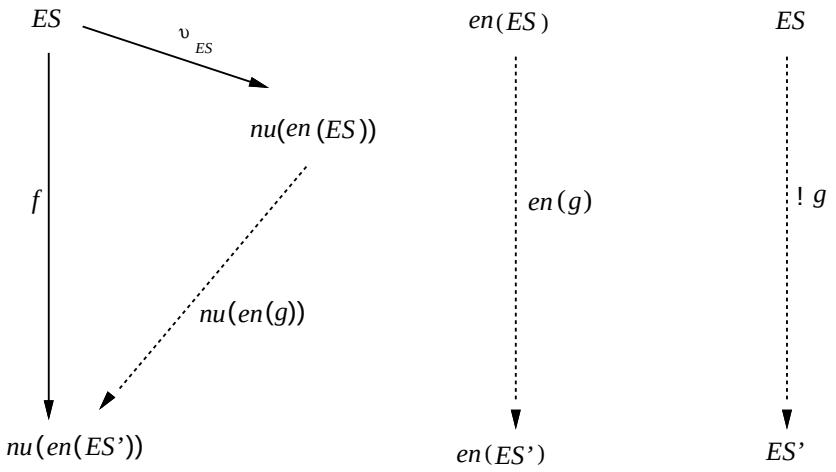
As a consequence of Theorem 5.7.10 each L-event structure can in fact be represented as an UL-event structure in a canonical way.

Corollary 5.7.11

$i : \mathcal{ULES} \to \mathcal{LES}$ and $nu \circ en : \mathcal{LES} \to \mathcal{ULES}$ form a co-reflection with i the left adjoint and the arrows υ_{ES} as unit.

Proof.

Let ES be an UL-event structure, let ES' be an L-event structure, and let f be an LES-morphism from ES to $nu(en(ES'))$. It must be proved that there is a unique LES-morphism g from ES to ES' such that the following diagram commutes.



By Theorem 5.7.10 there exists a unique PN-morphism (β, η) from $en(ES)$ to $en(ES')$ such that $nu((\beta, \eta)) \circ \upsilon_{ES} = f$. Then because en is full and faithful there

exists a unique LES-morphism g from ES to ES' such that $en(g) = (\beta, \eta)$ and hence $nu \circ en(g) \circ v_{ES} = f$. □

In the beginning of this section we argued that it is not possible to obtain a co-reflection between $\mathcal{ULES}$ and $\mathcal{PN}$. Hence we have restricted the category $\mathcal{PN}$ by cutting down on the *objects*. Another possibility is to cut down on the *arrows* of $\mathcal{PN}$ by considering only co-injective PN-morphisms.

From Lemma 2.2.2 we immediately have that if (β, η) is a co-injective PN-morphism from N_1 to N_2, then $\eta(\rho) \in SFS_{N_2}$ for all $\rho \in SFS_{N_1}$.

It is easy to see that the proof of the co-reflection between $\mathcal{ULES}$ and $\mathcal{PNS}$ still goes through with $\mathcal{PNC}$, the category of Petri nets with co-injective morphisms, instead of $\mathcal{PNS}$ (where nu is extended to a functor from $\mathcal{PNC}$ to $\mathcal{ULES}$ in the obvious way). Hence we also have the following result.

Theorem 5.7.12

$en \circ i : \mathcal{ULES} \to \mathcal{PNC}$ and $nu : \mathcal{PNC} \to \mathcal{ULES}$ form a co-reflection with en the left adjoint and the arrows v_{ES} as unit. □

5.8 Local Multiset Event Structures

The event structure semantics for Petri nets defined in Section 5.4 does not take into account possible auto-concurrency in a Petri net, because it is based on the set of step firing sequences of a Petri net rather than the set of multiset firing sequences. A consequence of this restriction is that we only have a co-reflection with the category of UL-event structures for the category of co-safe Petri nets.

In this section we discuss a possibility for generalizing the co-reflection between the category of UL-event structures and the category of co-safe Petri nets (Theorem 5.7.10) in order to obtain an event structure semantics for the category of *all* Petri nets. To this aim we define local event structures with *multisets* rather than sets (called LM-event structures) to take auto-concurrency into account, and show that there exists an adjunction between the category of LM-event structures satisfying the unique occurrence property (called ULM-event structures) and the category of Petri nets. The proof of this result proceeds along the same lines as the proof of Theorem 5.7.10 (even though notationally it is more involved), and hence most of the details are omitted.

The adjunction between the category of ULM-event structures and the category of Petri nets is however not a co-reflection. Moreover, we show that it is not possible to cut this adjunction down to a co-reflection by restricting the category of ULM-event structures.

Local event structures are generalized by allowing multisets of events in the set of configurations and the enabling relation.

Definition 5.8.1

A *local multiset event structure* is a triple $ES = (E, C, \vdash)$ where E is a set of events, $C \subseteq M_F(E)$ is a non-empty set of (finite) configurations, and $\vdash \subseteq C \times M_F(E)$ is an enabling relation satisfying the following axioms.

(E0') $\underline{0} \neq c \Rightarrow \exists e \in E.\,(c(e) > 0 \text{ and } c - e \vdash e)$.

(E1') $c \vdash \underline{0}$.

(E2') $c \vdash u \Rightarrow \forall v \leq u.\,(c \vdash v \text{ and } c + v \vdash u - v)$. □

From now on we refer to local multiset event structures as LM-event structures.

As for L-event structures, the unique occurrence property for LM-event structures is defined using an equivalence relation over prime intervals, but now with respect to multiset sequences.

So let X be a set and let $L \subseteq (M_F(X))^+$ be a set of multiset sequences satisfying the following two properties.

(L1') $\rho u \in L \Rightarrow \rho \in L$.

(L2') $\rho u \in L \Rightarrow \forall v \leq u.\,\rho v(u - v) \in L$.

The set of prime intervals of L is given by

$$PI_L = \{\rho a \mid \rho a \in L\}$$

and the function $int_L : L \to M_F(PI_L)$ mapping each element from L to the multiset of multiset prime intervals of L is given by:

- $int_L(\underline{0}) = \underline{0}$ and
- $int_L(\rho u) = int_L(\rho) + \sum_{a \in X, u(a) > 0} u(a) \cdot \rho a$.

The subscript $_L$ may again be omitted if L is clear from the context.

Let $R \subseteq PI \times PI$ be an equivalence relation. Then for $\rho a \in PI$,

$$\langle \rho a \rangle_R = \{\rho' a' \in PI \mid \rho' a' \, R \, \rho a\}.$$

The function $past_R : L \to M_F(PI/R)$ is given by:

$$past_R(\rho)(\langle \rho_1 a_1 \rangle_R) = \sum_{\rho_2 a_2 \in \langle \rho_1 a_1 \rangle_R} int(\rho)(\rho_2 a_2).$$

Note that if L is a set of step sequences, then these notions indeed coincide with the notions defined in Section 5.3.

Now R is said to be L-consistent iff it satisfies the following two conditions.

(C1") $(\rho u \in L \text{ and } u(a) > 0) \Rightarrow \rho a \, R \, \rho(u - a)a$.

(C2") $\rho a, \rho' a \in PI \Rightarrow (past_R(\rho) = past_R(\rho') \Rightarrow \rho a \, R \, \rho' a)$.

By the following lemma there exists also in this case a least equivalence relation which is L-consistent.

Lemma 5.8.2

Let X be an alphabet, let $L \subseteq (M_F(X))^+$ be a set of multiset sequences satisfying (L1') and (L2'), and let $K = \{R \subseteq PI \times PI \mid R \text{ is an } L\text{-consistent equivalence relation}\}$. Then $K \neq \emptyset$ and $\bigcap K \in K$.

Proof.

Because $PI \times PI \in K$ it is sufficient to prove that $\hat{R} = \bigcap K$ is an L-consistent equivalence relation.

It is clear that $\hat{R}$ is an equivalence relation which satisfies (C1").

Let $\rho a, \rho' a \in PI$ be such that $past_{\hat{R}}(\rho) = past_{\hat{R}}(\rho')$. It suffices to prove that $past_R(\rho) = past_R(\rho')$ for every $R \in K$.

So let $R \in K$ and suppose $\langle \rho_1 a_1 \rangle_R \in PI/R$. Then because $\hat{R} \subseteq R$ there exist $\rho_1^1 a_1^1, \ldots, \rho_1^k a_1^k \in PI$ for some $k \geq 1$ such that $\langle \rho_1^i a_1^i \rangle_{\hat{R}} \cap \langle \rho_1^j a_1^j \rangle_{\hat{R}} = \emptyset$ for all $1 \leq i, j \leq k$ with $i \neq j$ and $\bigcup_{i=1}^k \langle \rho_1^i a_1^i \rangle_{\hat{R}} = \langle \rho_1 a_1 \rangle_R$.

From the definition of *past* and the fact that $past_{\hat{R}}(\rho) = past_{\hat{R}}(\rho')$ we then have that $past_R(\rho)(\langle \rho_1 a_1 \rangle_R) = \sum_{\rho_2 a_2 \in \langle \rho_1 a_1 \rangle_R} int(\rho)(\rho_2 a_2) = \sum_{i=1}^k \sum_{\rho_2 a_2 \in \langle \rho_1^i a_1^i \rangle_{\hat{R}}} int(\rho)(\rho_2 a_2) = \sum_{i=1}^k past_{\hat{R}}(\rho)(\langle \rho_1^i a_1^i \rangle_{\hat{R}}) = \sum_{i=1}^k past_{\hat{R}}(\rho')(\langle \rho_1^i a_1^i \rangle_{\hat{R}}) = \sum_{i=1}^k \sum_{\rho_2 a_2 \in \langle \rho_1^i a_1^i \rangle_{\hat{R}}} int(\rho')(\rho_2 a_2) = \sum_{\rho_2 a_2 \in \langle \rho_1 a_1 \rangle_R} int(\rho')(\rho_2 a_2) = past_R(\rho')(\langle \rho_1 a_1 \rangle_R)$. □

Let $\approx_L \subseteq PI \times PI$ denote the least equivalence relation which is L-consistent.

The equivalence relation $\approx_L$ has the following properties which can be proved similar to the proof of Lemma 5.3.4

Lemma 5.8.3

Let X be an alphabet, let $L \subseteq (M_F(X))^+$ be a set of multiset sequences satisfying (L1') and (L2'), and let $\rho_1 a_1, \rho_2 a_2 \in PI$ be such that $\rho_1 a_1 \approx_L \rho_2 a_2$. Then

(1) $a_1 = a_2$

(2) $\rho_1 a_1 \approx_{L'} \rho_2 a_2$ for every $L' \subseteq (M_F(X))^+$ satisfying (L1') and (L2') with $L \subseteq L'$. □

Let $ES = (E, C, \vdash)$ be an LM-event structure. Then $MFS_{ES} \subseteq (M_F(E))^+$, the set of *multiset firing sequences* of ES, and $cf_{ES} : MFS_{ES} \to M_F(E)$, the function which maps each multiset firing sequence to the configuration it leads to, are defined inductively as:

- $\underline{0} \in MFS_{ES}$ and $cf_{ES}(\underline{0}) = \underline{0}$
- $(\rho \in MFS_{ES}$ and $cf_{ES}(\rho) \vdash u) \Rightarrow (\rho u \in MFS_{ES}$ and $cf_{ES}(\rho u) = cf_{ES}(\rho) + u)$.

Note that $cf_{ES}(\rho) = mset(\rho)$ for every $\rho \in MFS_{ES}$.

The set MFS_{ES} yields an equivalence relation $\approx_{MFS_{ES}}$, which is denoted by $\approx_{ES}$ in what follows.

Then ES has the unique occurrence property iff it satisfies the following two conditions.

(U1') $\forall e \in E. \exists \rho e \in PI_{ES}$

(U2') $\forall \rho_1 e, \rho_2 e \in PI_{ES}. \rho_1 e \approx_{ES} \rho_2 e$.

LM-event structures satisfying the unique occurrence property will be referred to as ULM-event structures.

Associated with every Petri net N we have an equivalence relation $\approx_{MFS_N}$, from now on denoted by $\approx_N$. Similar to Section 5.4 this equivalence relation is used for defining a map nm which associates an LM-event structure $nm(N)$ with every Petri net N.

Definition 5.8.4
Let $N = (S, T, W, M_{in})$ be a Petri net. Then $nm(N) = (E, C, \vdash)$ where

- $E = \{\langle \rho t \rangle_N \mid \rho t \in PI_N\}$
- $C = \{past_N(\rho) \mid \rho \in MFS\}$
- $\vdash \subseteq C \times M_F(E)$ is given by:

$$c \vdash u \Leftrightarrow \exists \rho v \in MFS.\ past_N(\rho) = c \text{ and } u = past_N(\rho v) - past_N(\rho).$$

□

Theorem 5.8.5
Let $N = (S, T, W, M_{in})$ be a Petri net. Then $nm(N)$ is an ULM-event structure.

Proof.
Similar to the proof of Theorem 5.4.7. □

The map nu defined in Section 5.4 associates an UL-event structure with every Petri net N based on SFS. Because we now use MFS rather than SFS, the map nm yields for a Petri net N which is not co-safe, an event structure different from $nu(N)$.

Example 5.8.6
For the Petri net N_2 depicted in Figure 2.2, its associated L- and LM-event structure are depicted in Figure 5.9. Since N_2 is not co-safe, $nu(N_2)$ and $nm(N_2)$ are different.

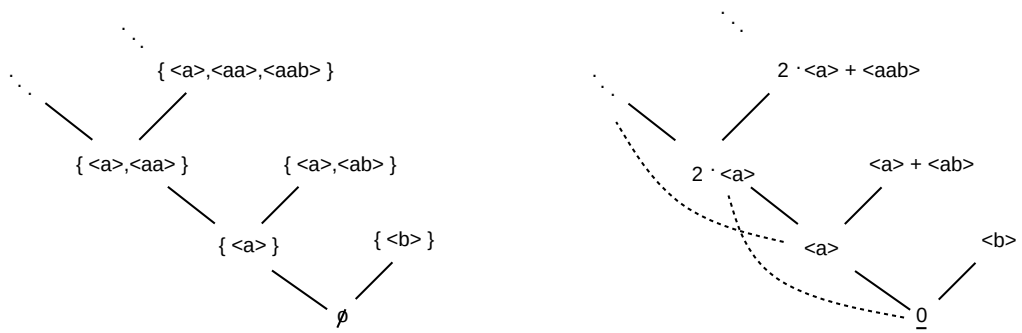


Figure 5.9: The L-event structure $nu(N_2)$ and the LM-event structure $nm(N_2)$

In $nu(N_2)$ there is an infinite number of events corresponding to a. In $nm(N_2)$ however, all prime intervals corresponding to a are equivalent due to auto-concurrency. Hence $nm(N_2)$ has only one event corresponding to a. In both $nu(N_2)$ and $nm(N_2)$ there is an infinite number of events corresponding to b. □

The main reason why the map nu from Section 5.4 cannot be extended to a functor from $\mathcal{PN}$ to $\mathcal{ULES}$ is that prime intervals with respect to *SFS* and equivalence of prime intervals are not preserved under PN-morphisms. Prime intervals with respect to *MFS* however, are preserved under PN-morphisms by Lemma 2.2.2. This enables us to extend nm to a functor from $\mathcal{PN}$ to the category of ULM-event structures.

Definition 5.8.7

Let $\mathcal{LMES}$ be the category which has LM-event structures as its objects and *LMES-morphisms* as its arrows.

An LMES-morphism from an LM-event structure $(E_1, C_1, \vdash_1)$ to an LM-event structure $(E_2, C_2, \vdash_2)$ is a partial function $f : E_1 \to E_2$ such that:

$$\forall c \in C_1.\, \forall u \in M_F(E_1).\, c \vdash_1 u \Rightarrow f(c) \vdash_2 f(u).$$

The identity morphism associated with an object is the identity function on its events; composition of LMES-morphisms is composition of partial functions.

Let $\mathcal{ULMES}$ be the full subcategory of $\mathcal{LMES}$ the objects of which are ULM-event structures. □

Given a PN-morphism (β, η) from N_1 to N_2 where $nm(N_1) = (E_1, C_1, \vdash_1)$ and $nm(N_2) = (E_2, C_2, \vdash_2)$, define $nm((\beta, \eta)) = f$ with $f : E_1 \to E_2$ given by: $f(\langle \rho t \rangle_{N_1})$ is undefined if $\eta(t)$ is undefined and $f(\langle \rho t \rangle_{N_1}) = \langle \eta(\rho)\eta(t) \rangle_{N_2}$ otherwise for all $\langle \rho t \rangle_{N_1} \in E_1$.

In the same way as in Section 5.7 it can be proved that this map is well-defined and yields a functor.

Theorem 5.8.8

nm is a functor from $\mathcal{PN}$ to $\mathcal{ULMES}$. □

Example 5.8.9

Let (β, η) be the PN-morphism from N to N' from Figure 2.7. As argued in Example 5.7.2, there is no corresponding LES-morphism from the UL-event structure $nu(N)$ to the UL-event structure $nu(N')$. In Figure 5.10 this PN-morphism is depicted, together with the ULM-event structures $nm(N)$ and $nm(N')$. The LMES-morphism $nm((\beta, \eta))$ maps both $\langle a \rangle_N$ and $\langle b \rangle_N$ to $\langle c \rangle_{N'}$. □

It is also straightforward to lift the functor en from $\mathcal{LES}$ to $\mathcal{PNS}$ to a functor mn from $\mathcal{LMES}$ to $\mathcal{PN}$.

Given an LM-event structure $ES = (E, C, \vdash)$, define the multiset transition system $mt(ES) = (C, E, \longrightarrow_{ES}, \underline{0})$ where

$$c \xrightarrow{u}_{ES} c' \Leftrightarrow (c \vdash u \text{ and } c' = c + u).$$

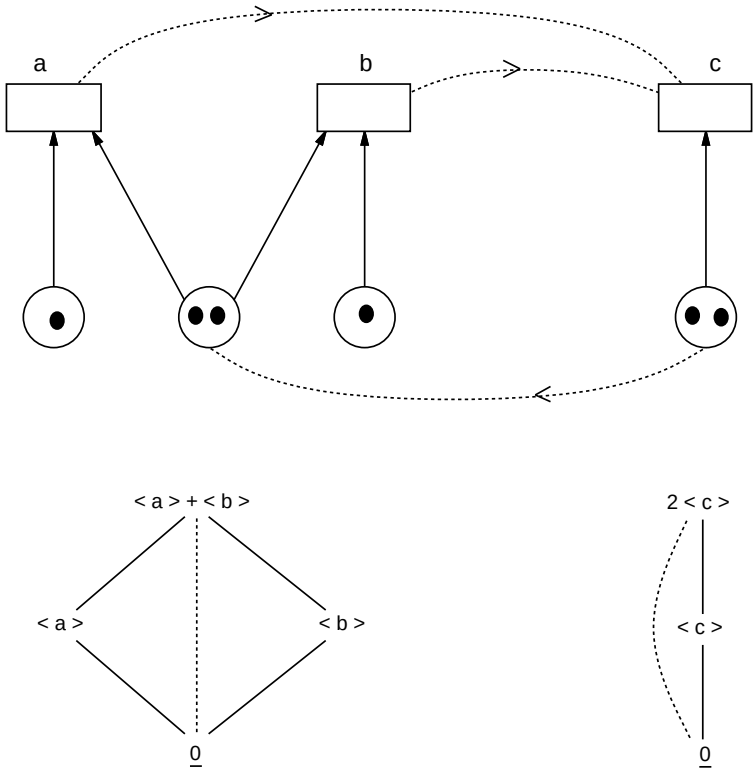


Figure 5.10: The ULM-event structures $nm(N)$ and $nm(N')$

Then define $mn(ES) = tn(mt(ES))$.

The map en associates a Petri net with every L-event structure which has the same set of step firing sequences. For the map mn we only have the following inclusion which follows easily from the definition of mn.

Lemma 5.8.10

Let $ES = (E, C, \vdash)$ be an LM-event structure. Then $MFS_{ES} \subseteq MFS_{mn(ES)}$. □

Example 5.8.11

Consider the LM-event structure ES_7 depicted in Figure 5.11. Note that ES_7 is even an ULM-event structure.

Because there is no upperbound on the number of occurrences of a, ${}^ra \leq a^r$ for every region r of ES_7. Thus in $mn(ES_7)$, the transition b cannot be disabled by the occurrence of a. It is easy to see that $mn(ES_7)$ is the saturated version of the Petri net N_2 depicted in Figure 2.2, and hence $MFS_{mn(ES_7)} = MFS_{N_2}$. Thus MFS_{ES_7} is properly included in $MFS_{mn(ES_7)}$. □

Let $ES = (E, C, \vdash)$ be an ULM-event structure with $nm(mn(ES)) = (\hat{E}, \hat{C}, \hat{\vdash})$. Again we can define, using Lemma 5.8.3(2) and Lemma 5.8.10, a map $v_{ES} : E \to \hat{E}$ by: $v_{ES}(e) = \langle \rho e \rangle_{mn(ES)}$ where $\rho e \in PI_{ES}$. As Example 5.8.11, together with Example 5.8.6, illustrates, this map is not an LMES-isomorphism as in Lemma 5.5.4.

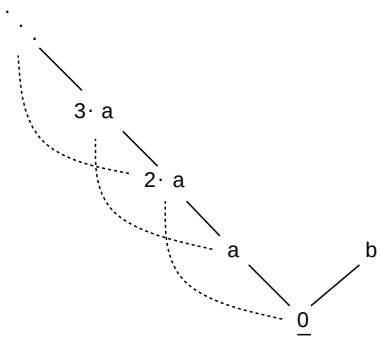


Figure 5.11: The LM-event structure ES_7

We now only have the following result, which can be proved in the same way as Lemma 5.5.4.

Lemma 5.8.12
Let ES be an ULM-event structure. Then v_{ES} is an LMES-morphism from ES to $nm(mn(ES))$. □

In order to extend mn to a functor, define given an LMES-morphism f from $ES_1 = (E_1, C_1, \vdash_1)$ to $ES_2 = (E_2, C_2, \vdash_2)$, $mt(f) = (f, g)$ with $g : C_1 \to C_2$ given by $g(c) = f(c)$. Now define $mn(f) = tn(mt(f))$.

Similar to the proof of Theorem 5.7.7 the following can now be proved.

Theorem 5.8.13
mn is a faithful functor from $\mathcal{LMES}$ to $\mathcal{PN}$. □

As the following example illustrates the functor mn, even when restricted to $\mathcal{ULMES}$, is however not full.

Example 5.8.14
Consider the ULM-event structures ES_8 and ES_9 depicted in Figure 5.12.

It is easy to see that $mn(ES_8)$ and $mn(ES_9)$ are the same. Then there is a PN-morphism from $mn(ES_8)$ to $mn(ES_9)$, the identity PN-morphism, for which there is no corresponding LMES-morphism from ES_8 to ES_9. □

We do however still have the following result.

Theorem 5.8.15
$mn : \mathcal{ULMES} \to \mathcal{PN}$ and $nm : \mathcal{PN} \to \mathcal{ULMES}$ form an adjunction with mn the left adjoint and the arrows v_{ES} as unit.

Proof.
Similar to the proof of Theorem 5.7.10. □

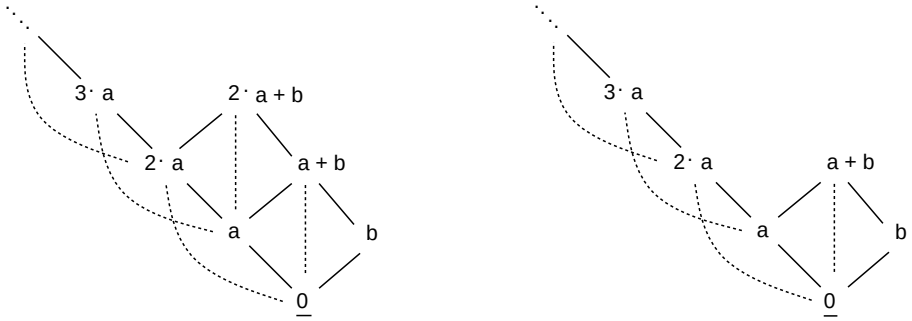


Figure 5.12: ULM-event structures ES_8 and ES_9 yielding the same Petri net

As we mentioned before, the arrows v_{ES} are not LMES-isomorphisms, and hence the adjunction is not a co-reflection. As the following example illustrates, the arrows v_{ES} are not even LMES-isomorphisms for ULM-event structures ES for which there exists a Petri net N with $nm(N) = ES$. Hence it is not possible to cut the adjunction down to a co-reflection by restricting the category of ULM-event structures.

Example 5.8.16
Consider the ULM-event structure $ES = nm(N_2)$ from Example 5.8.6. Then ${}^r\langle a\rangle_{ES} \leq \langle a\rangle_{ES}{}^r$ for every region r of ES. Hence in the Petri net $mn(ES)$ the transition $\langle b\rangle_{ES}$ cannot be disabled by the occurrence of $\langle a\rangle_{ES}$. Then it is easy to see that v_{ES} is not an LMES-isomorphism. □

It might be possible to solve this problem by strengthening the equivalence relation $\approx_L$. This should be done in such a way that, for the Petri net N_2 from Example 5.8.6, all prime intervals corresponding to b are identified. Then it could be that the adjunction between $\mathcal{ULMES}$ and $\mathcal{PN}$ can be cut down to a co-reflection by restricting the objects of $\mathcal{ULMES}$ to ULM-event structures satisfying some "regional" axioms. It is however not clear at present how all this can be achieved.

A consequence of the co-reflection between $\mathcal{ULES}$ and $\mathcal{PNS}$ is that $\mathcal{ULES}$ is a co-reflective subcategory of $\mathcal{LES}$. From Example 5.8.14 it easily follows that we do not have a similar result in the present setting due to the fact that mn is not full.

5.9 Concluding Remarks

In Section 5.1 prime event structures have been used to give an event structure semantics for 1-safe Petri nets, following the approach from [98]. Prime event structures have also been proposed as possible candidates for representing the behaviour of general Petri nets [61] (see also [24]). In this approach general Petri nets lead to the same class of event structures as 1-safe Petri nets by viewing the tokens more or less as "coloured" entities.

In this chapter we have proposed a *proper* generalization of the prime event structure semantics for 1-safe Petri nets. Our event structure semantics in terms of L-event structures is however restricted in the sense that auto-concurrency is filtered out from the behaviour of Petri nets. For the objects in the category $\mathcal{PN}s$ of 1-safe Petri nets our event structure semantics agrees with the prime event structure semantics from [66, 98]. In this chapter we have however restricted this comparison to the level of objects, due to the (slight) differences between our PN-morphisms and the Petri net morphisms used in [98].

For 1-safe Petri nets there exists by Lemma 5.1.3 and Lemma 5.1.6 a bijection between the finite configurations of its associated prime event structure and the M-traces of its associated M-trace language. Moreover, the posets obtained by equipping M-traces with their M-trace ordering relation and by equipping the finite configurations of the prime event structure with the inclusion ordering are isomorphic.

For general Petri nets on the other hand, there is in general no bijection between the finite configurations of its associated UL-event structure and the L-traces of its associated L-trace language, because step firing sequences which have the same past are not necessarily in the same L-trace. This is for instance the case for the Petri net N_{10} depicted in Figure 5.4 for which $ab \not\cong_{N_{10}} ba$, but $past_{N_{10}}(ab) = past_{N_{10}}(ba)$. This has led us to introduce a new equivalence relation over prime intervals.

The definition of the unique occurrence property of L-event structures is based on this equivalence relation over prime intervals. These prime intervals are defined in terms of the step firing sequences of the L-event structure. One might now wonder if it is necessary to take into account the whole history of event occurrences, as is done in both prime intervals and condition (C2). In [66] for instance, prime intervals of partial orders are defined which only consist of pairs of elements such that one "covers" the other. The equivalence relation over these prime intervals, which is used in [66] to extract event occurrences from the partial order, then simply identifies prime intervals which are connected by diamonds. This suggests, in the context of L-event structures, also a simpler formulation of the unique occurrence property in the following way.

Let $ES = (E, C, \vdash)$ be an L-event structure. Then let $PI'_{ES} \subseteq C \times E$ be the set of *C-prime intervals* of ES, given by:

$$PI'_{ES} = \{(c, e) \in C \times E \mid c \vdash e\}.$$

The equivalence relation $\approx'_{ES} \subseteq PI'_{ES} \times PI'_{ES}$ is defined as the least equivalence relation such that:

$$(c \vdash u \text{ and } e \in u) \Rightarrow (c, e) \approx'_{ES} (c \cup (u - e), e).$$

Then $ES = (E, C, \vdash)$ has the *unique C-occurrence property* iff it satisfies the following conditions (U1") and (U2").

(U1") $\forall e \in E. \exists (c, e) \in PI'_{ES}$

(U2") $\forall (c_1, e), (c_2, e) \in PI'_{ES}. (c_1, e) \approx'_{ES} (c_2, e)$.

The class of L-event structures satisfying this C-unique occurrence property is however not the same as the class of UL-event structures. The reason is that $\approx'_{ES}$ identifies event occurrences which are distinguished by $\approx_{ES}$.

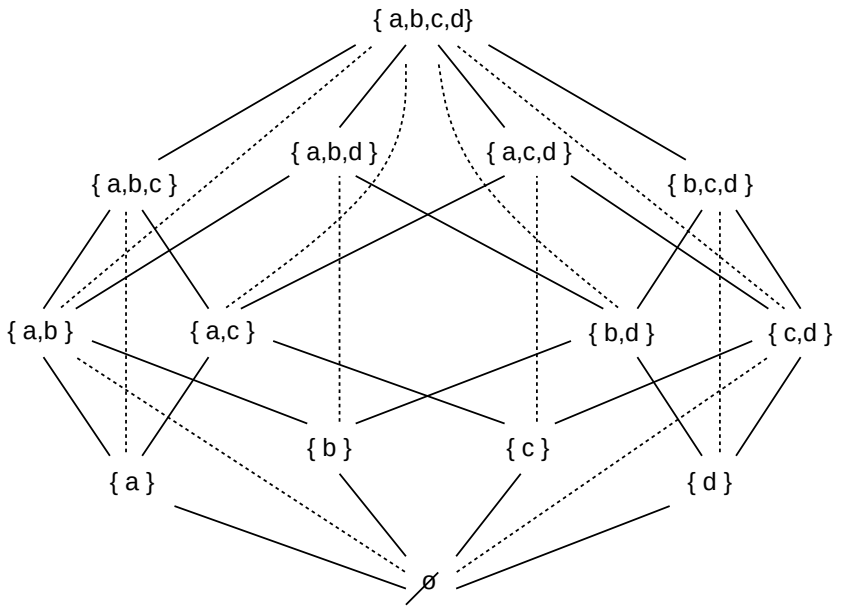


Figure 5.13: The L-event structure ES_{10}

To see this, consider the L-event structure ES_{10} depicted in Figure 5.13. In ES_{10}, all diamonds represent concurrency, except that $\neg(\emptyset \vdash \{a,c\})$ and $\neg(\emptyset \vdash \{b,d\})$. For each event there is exactly one equivalence class of C-prime intervals under $\approx'_{ES_{10}}$. For instance, for the event a we have:

$$(\emptyset, a) \approx'_{ES_{10}} (\{b\}, a) \approx'_{ES_{10}} (\{b,d\}, a) \approx'_{ES_{10}} (\{b,c,d\}, a) \approx'_{ES_{10}} (\{c,d\}, a) \approx'_{ES_{10}} (\{c\}, a).$$

Hence ES_{10} has the unique C-occurrence property.

On the other hand, ES_{10} does not have the unique occurrence property with respect to $\approx_{ES_{10}}$, because for each event there are two equivalence classes: one containing its prime intervals starting with a, b, or $\{a,b\}$, and one containing its prime intervals starting with c, d, or $\{c,d\}$. So with respect to $\approx_{ES_{10}}$, there are two distinguishable occurrences of a in this L-event structure, $\langle a \rangle_{ES_{10}}$ and $\langle ca \rangle_{ES_{10}}$. Similarly, there are also two distinguishable occurrences of the other events.

At present it is not clear, if with the alternative definition of the unique occurrence property also a co-reflection with the category of co-safe Petri nets can be obtained.

Chapter 6

A Categorical Classification of Event Structures

In Chapter 5 we have introduced L-event structures and UL-event structures in our proposal for lifting the prime event structure semantics for 1-safe Petri nets to the level of general Petri nets. In this chapter we investigate in detail the relationship between the new (U)L-event structures and some basic classes of event structures which have appeared in the literature.

In Section 5.1 the prime event structures from [66] have been introduced as sets of events together with a causal dependency relation and a binary conflict relation. In this chapter we also consider the generalization of prime event structures used in [96] in which the conflict relation is replaced by a consistency predicate to express that there may be arbitrary conflicts between events. These event structures are here called *G-prime event structures*.

In addition to (G-)prime event structures we also consider Winskel's general event structures [96] (see also [92]) and their stable subclass. These general event structures are obtained by further generalizing G-prime event structures by replacing the causal dependency relation by a (global) enabling relation specifying the enabling of events at consistent sets. In this thesis we refer to these event structures as *W-event structures*. Also W-event structures lead to a notion of a configuration. In [96] it has been proved that an equivalent specification of W-event structures is obtained by means of families of configurations. In fact, it is sufficient to consider only the finite configurations because the set of all configurations can be derived from the set of finite configurations through a standard construction known as ideal completion.

For W-event structures there is no global causal dependency relation over the events. There is not even a causal dependency relation for the events in a given configuration. This led in [96, 92] to the notion of a *stable* W-event structure in which each event in a configuration is enabled by a unique minimal set of events.

The comparisons between these various classes are performed in a categorical framework by exhibiting adjunctions between the various categories. Recall that in Corollary 5.7.11 it has already been proved that the category $\mathcal{ULES}$ of UL-event structures is a full co-reflective subcategory of the category $\mathcal{LES}$ of L-event structures.

In Section 6.1 we prove that the relationship between the category $\mathcal{LES}$ of L-

event structures and the category $\mathcal{WES}$ of W-event structures can be expressed as a reflection, with the left adjoint going from $\mathcal{LES}$ to $\mathcal{WES}$. This means that $\mathcal{WES}$ can be viewed as being embedded in $\mathcal{LES}$. A characterization is given of the full subcategory of $\mathcal{LES}$ for which the reflection cuts down to an equivalence.

We then show in Section 6.2 that the reflection between $\mathcal{LES}$ and $\mathcal{WES}$ can be further extended to be a reflection between $\mathcal{LES}$ and $\mathcal{SWES}$, the category of stable W-event structures. Also for this reflection a characterization is given of the full subcategory of $\mathcal{LES}$ for which the reflection cuts down to an equivalence.

In Section 6.3 it is shown that a similar reflective relationship can also be established between $\mathcal{ULES}$ and $\mathcal{GPES}$, the category of G-prime event structures. The axioms needed on UL-event structures for cutting this reflection down to an equivalence are the same as for cutting down the reflection between $\mathcal{LES}$ and $\mathcal{SWES}$ to an equivalence. This result is closely related to the well-known fact that stable W-event structures and G-prime event structures correspond to the same class of Scott Domains.

Finally, we show in Section 6.4 that there is also a reflection between $\mathcal{ULES}$ and $\mathcal{PES}$, the category of prime event structures. The corresponding functor from $\mathcal{PES}$ to $\mathcal{ULES}$ is an extension of the map *pu* defined in Section 5.4.

Finally in Section 6.5 we have some concluding remarks.

6.1 L-Event Structures and W-Event Structures

The *W-event structures* defined now are the general event structures from [96]. They are defined as families of configurations.

Definition 6.1.1

$\mathcal{WES}$ is the category of *W-event structures* specified as follows.
An object of $\mathcal{WES}$ is a W-event structure $W = (E, C)$ where E is a set of events and $C \subseteq P_F(E)$ is a non-empty set of (finite) configurations such that

(W1) $\emptyset \neq c \Rightarrow \exists e \in c.\, c - e \in C$

(W2) $c \uparrow c' \Rightarrow c \cup c' \in C$.

An arrow of $\mathcal{WES}$ is a *WES-morphism* $f : (E_1, C_1) \rightarrow (E_2, C_2)$ which is a partial function $f : E_1 \rightarrow E_2$ such that

(1) $\forall c \in C_1.\, f(c) \in C_2$

(2) $\forall c \in C_1. \forall e_1, e_2 \in c.\, ((f(e_1) \text{ and } f(e_2) \text{ are defined and } f(e_1) = f(e_2)) \Rightarrow e_1 = e_2)$.

The identity morphism associated with an object is the identity function on its events and composition of arrows is composition of partial functions. □

A W-event structure is depicted through its configuration structure, which is the Hasse diagrams of its configurations, ordered under inclusion. In contrast to the

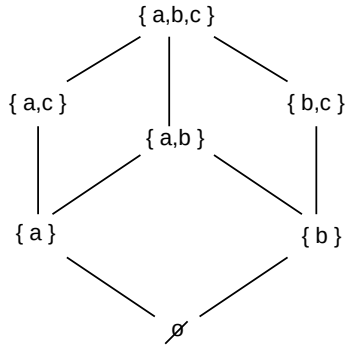


Figure 6.1: The W-event structure W_1

situation for L-event structures, for W-event structures this inclusion relation does carry an adequate amount of causal information.

In Figure 6.1 a W-event structure W_1 is depicted, which represents the "parallel switch" from [96].

The map we which is defined next yields for each W-event structure an L-event structure by interpreting diamonds in the configuration structure as concurrency to be expressed in the enabling relation.

For a W-event structure $W = (E, C)$, define $we(W) = (E, C, \vdash)$ where $\vdash \subseteq C \times P_F(E)$ is given by:

$$c \vdash u \Leftrightarrow c \cap u = \emptyset \text{ and } \forall v \subseteq u.\, c \cup v \in C.$$

Lemma 6.1.2

Let W be a W-event structure. Then $we(W)$ is an L-event structure.

Proof.

Follows easily from the definitions. □

For the W-event structure W_1 from Figure 6.1, the L-event structure $we(W_1)$ is depicted in Figure 6.2.

Clearly not every L-event structure is the image of a W-event structure. First of all, in contrast to W-event structures the diamonds in the configuration structure of an L-event structure do not necessarily represent concurrency. (This is for instance the case for the L-event structures ES_1 and ES_3 depicted in Figure 5.2)). Secondly, two configurations of an L-event structure may be compatible without their union being a configuration. This is for instance the case for the UL-event structure ES_{11} depicted in Figure 6.3.

We extend we to morphisms, by defining $we(f) = f$ for each WES-morphism f.

Lemma 6.1.3

Let f be a WES-morphism from $W_1 = (E_1, C_1)$ to $W_2 = (E_2, C_2)$. Then $we(f)$ is an LES-morphism from $we(W_1) = (E_1, C_1, \vdash_1)$ to $we(W_2) = (E_2, C_2, \vdash_2)$.

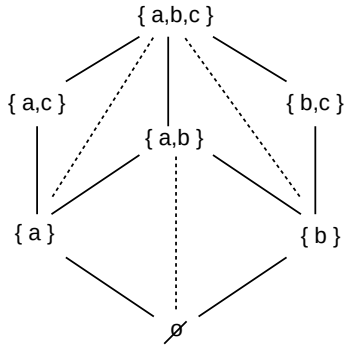


Figure 6.2: The L-event structure $we(W_1)$

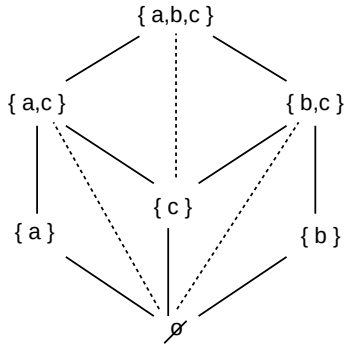


Figure 6.3: The UL-event structure ES_{11}

Proof.

Suppose that $c \vdash_1 u$. Then $c \cap u = \emptyset$ and $c \cup u \in C$. Hence $f(c) \cap f(u) = \emptyset$ by condition (2) in the definition of WES-morphism. Moreover, $c \cup v \in C_1$ for all $v \subseteq u$ and so by condition (1), $f(c \cup v) = f(c) \cup f(v) \in C_2$ for all $v \subseteq u$. Hence $f(c) \vdash_2 f(u)$. □

Lemma 6.1.2 and Lemma 6.1.3 lead to the following result.

Theorem 6.1.4

we is a functor from $\mathcal{WES}$ to $\mathcal{LES}$. □

Conversely, a map from L-event structures to W-event structures can be defined by dropping the local enabling relation representing concurrency, and closing the set of configurations with respect to union of compatible (with respect to inclusion) configurations.

This leads to the following definition of the map *ew* from L-event structures to W-event structures.

For an L-event structure $ES = (E, C, \vdash)$, define $ew(ES) = (E, \hat{C})$ where $\hat{C}$ is the least subset of $P_F(E)$ containing C which satisfies (W2).

Note that $ew(ES)$ is well-defined, because both $P_F(E)$ and $\bigcap\{C' \subseteq P_F(E) \mid C \subseteq C'$ and C' satisfies (W2)$\}$ satisfy (W2).

Lemma 6.1.5

Let $ES = (E, C, \vdash)$ be an L-event structure. Then $ew(ES) = (E, \hat{C})$ is a W-event structure.

Proof.

In order to prove that $ew(ES)$ satisfies (W1), let $\emptyset \neq c \in \hat{C}$. If $c \in C$, then there exists $e \in E$ such that $c - e \vdash e$ because ES satisfies (E0). Hence $c - e \in C \subseteq \hat{C}$. So assume that $c \notin C$. Then by the minimality of $\hat{C}$ there exist $c_1, c_2 \in \hat{C}$ with $c_1 \uparrow c_2$ such that $c = c_1 \cup c_2$, $|c_1| < |c|$, and $|c_2| < |c|$. Thus $|c| \geq 2$. Assume that for all $\hat{c} \in \hat{C}$ with $1 \leq |\hat{c}| < |c|$, there exists an $e \in E$ such that $\hat{c} - e \in \hat{C}$. Then there exist $e_1, \ldots, e_n \in E$ with $n = |c_1|$ such that $c_1 = \{e_1, \ldots, e_n\}$, and $\{e_1, \ldots, e_i\} \in \hat{C}$ for all $0 \leq i \leq n$. Because $|c_1| < |c|$ and $|c_2| < |c|$ there exists a largest integer k such that $k \in \{1, \ldots, n\}$ and $e_k \notin c_2$. Hence $e_{k+1}, \ldots, e_n \in c_2$. Then, by the definition of $\hat{C}$, $\{e_1, \ldots, e_{k-1}\} \cup c_2 = c - e_k \in \hat{C}$. This proves that $ew(ES)$ satisfies (W1).

From the definition of $ew(ES)$ we immediately have that $ew(ES)$ satisfies (W2). □

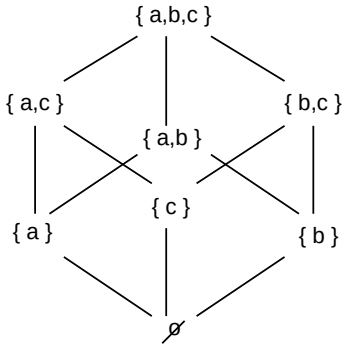


Figure 6.4: The W-event structure $ew(ES_{11})$

For the L-event structure ES_{11} depicted in Figure 6.3, the W-event structure $ew(ES_{11})$ is depicted in Figure 6.4.

The definition of ew implies that the set of configurations of an L-event structure is contained in the set of configurations of its image under ew. As the next lemma shows, all configurations added by ew are subsets of original configurations. This observation is used when we extend ew to a functor.

Lemma 6.1.6

Let $ES = (E, C, \vdash)$ be an L-event structure with $ew(ES) = (E, \hat{C})$. Then $\hat{c} \in \hat{C}$ implies that there exists $c \in C$ such that $\hat{c} \subseteq c$.

Proof.

Let $\hat{c} \in \hat{C}$. If $\hat{c} \in C$ then the claim holds trivially, so suppose that $\hat{c} \in \hat{C} - C$. Now assume to the contrary that there exists no $c \in C$ such that $\hat{c} \subseteq c$. Let $C' = \hat{C} - \{c' \in \hat{C} \mid \hat{c} \subseteq c'\}$. Then $C \subseteq C'$ because $C \subseteq \hat{C}$ and $\{c' \in \hat{C} \mid \hat{c} \subseteq c'\} \cap C = \emptyset$. Suppose $c_0, c_1, c_2 \in C'$ are such that $c_1 \subseteq c_0$ and $c_2 \subseteq c_0$. $\hat{C}$ satisfies (W2) and so $c_1 \cup c_2 \in \hat{C}$. By $c_1 \cup c_2 \subseteq c_0 \in C'$ and $\hat{c} \not\subseteq c_0$ we have $\hat{c} \not\subseteq c_1 \cup c_2$. Hence $c_1 \cup c_2 \in C'$. This leads to the conclusion that C' satisfies (W2), a contradiction with the minimality of $\hat{C}$. Thus there exists $c \in C$ such that $\hat{c} \subseteq c$. □

Now define $ew(f) = f$ for each LES-morphism f.

Lemma 6.1.7

Let f be an LES-morphism from $ES_1 = (E_1, C_1, \vdash_1)$ to $ES_2 = (E_2, C_2, \vdash_2)$. Then $ew(f)$ is a WES-morphism from $ew(ES_1) = (E_1, \hat{C}_1)$ to $ew(ES_2) = (E_2, \hat{C}_2)$.

Proof.

Let $c \in \hat{C}_1$. By condition (1) in the definition of WES-morphism, $f(c) \in \hat{C}_2$ should hold. If $c \in C_1$, then by condition (E1) in the definition of an L-event structure, $c \vdash_1 \emptyset$. Since f is an LES-morphism, we have in this case $f(c) \vdash_2 \emptyset$ and so $f(c) \in C_2 \subseteq \hat{C}_2$. Using this observation we now prove by induction on $|c|$ that $f(c) \in f(\hat{C}_2)$ always holds. If $|c| = 0$ or $|c| = 1$, then $c \in \hat{C}_1$ and we are done. Now assume that $|c| > 1$ with $c \in \hat{C}_1 - C_1$. Then by the minimality of $\hat{C}_1$ there exist $c_1, c_2 \in \hat{C}_1$ such that $c = c_1 \cup c_2$, $|c_1| < |c|$, and $|c_2| < |c|$. Hence $f(c_1), f(c_2) \in \hat{C}_2$ by the induction hypothesis. By Lemma 6.1.6 there exists a $c' \in C_1$ such that $c \subseteq c'$. We then have as above that $f(c') \in C_2 \subseteq \hat{C}_2$. Thus $f(c_1), f(c_2), f(c') \in \hat{C}_2$ and $f(c_1) \subseteq f(c')$ and $f(c_2) \subseteq f(c')$. Then $f(c_1) \cup f(c_2) = f(c) \in \hat{C}_2$ because $\hat{C}_2$ satisfies (W2).

That condition (2) in the definition of a WES-morphism is satisfied by f can be seen as follows: let $c \in \hat{C}_1$ and $e_1, e_2 \in c$ be such that $e_1 \neq e_2$ and $f(e_1)$ and $f(e_2)$ are both defined. Again Lemma 6.1.6 guarantees the existence of a $c' \in C_1$ such that $c \subseteq c'$. Then Lemma 5.2.3(1) gives $f(e_1) \neq f(e_2)$. □

Lemma 6.1.5 and Lemma 6.1.7 yield the following result.

Theorem 6.1.8

ew is a functor from $\mathcal{LES}$ to $\mathcal{WES}$. □

The functors ew and we are now shown to form an adjunction with as co-unit the identity arrows id_W. Hence this adjunction is a reflection.

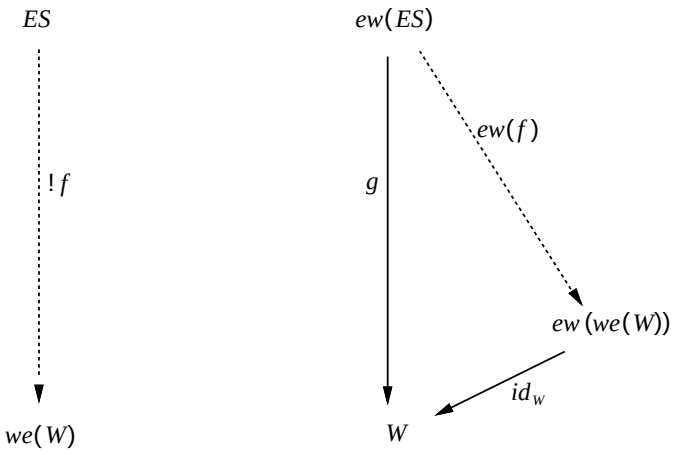
Theorem 6.1.9

$ew : \mathcal{LES} \to \mathcal{WES}$ and $we : \mathcal{WES} \to \mathcal{LES}$ form a reflection with ew the left adjoint and the identity arrows id_W as co-unit.

Proof.

First note that the co-unit is well-defined because $ew(we(W)) = W$ for each W-event structure W.

Let $ES = (E, C, \vdash)$ be an L-event structure, let $W = (E', C')$ be a W-event structure, and let g be a WES-morphism from $ew(ES) = (E, \hat{C})$ to W. Then we must prove that there exists a unique LES-morphism f from ES to $we(W) = (E', C', \vdash')$ such that the following diagram commutes.



Since ew is the identity on arrows, it is sufficient to prove that g is an LES-morphism from ES to $we(W)$. Suppose $c \vdash u$. Then $c \cap u = \emptyset$ and $c \cup v \in C$, for all $v \subseteq u$ by (E2). Since g is a WES-morphism from $ew(ES)$ to W we now have that $c \cup v \in C \subseteq \hat{C}$ implies $g(c) \cup g(v) \in C'$, for all $v \subseteq u$, and $g(c) \cap g(u) = \emptyset$. Hence $g(c) \vdash' g(u)$. □

Finally we formulate two conditions on L-event structures which can be used to identify a full subcategory of $\mathcal{LES}$ for which the reflection in Theorem 6.1.9 cuts down to an equivalence.

Let $(E, C, \vdash)$ be an L-event structure. Then (FC) and (D) are defined as follows.

(FC) $\forall c, c' \in C.\, (c \uparrow c' \Rightarrow c \cup c' \in C)$.

(D) $(c \cap u = \emptyset \text{ and } \forall v \subseteq u.\, c \cup v \in C) \Rightarrow c \vdash u$.

The axiom (FC) is a "forward closure" property stating that configurations which are compatible (with respect to inclusion) can be joined. The axiom (D) states that all "diamonds" represent concurrency.

Clearly, for every W-event structure W, the L-event structure $we(W)$ satisfies (FC) and (D). It is also easy to see that for every L-event structure ES satisfying (FC) and (D), $ES \equiv we(ew(ES))$. Hence we have the following corollary of Theorem 6.1.9.

Corollary 6.1.10

$\mathcal{WES}$ is equivalent to the full subcategory of $\mathcal{LES}$ the objects of which are L-event structures satisfying (FC) and (D). □

6.2 L-Event Structures and Stable W-Event Structures

Stable W-event structures [96] are W-event structures satisfying an additional axiom guaranteeing that each event is enabled by a unique minimal set of events.

Definition 6.2.1
$\mathcal{SWES}$, the category of *stable W-event structures*, is the full subcategory of $\mathcal{WES}$ the objects (E, C) of which satisfy

(W3) $c \uparrow c' \Rightarrow c \cap c' \in C$. □

The W-event structure W_1 depicted in Figure 6.1 is an example of a W-event structure which is not stable: $\{a, c\} \uparrow \{b, c\}$, but not $\{c\} \in C$. The W-event structure $ew(ES_{11})$ depicted in Figure 6.4 is stable.

In this section we prove that there is also a reflection between $\mathcal{LES}$ and $\mathcal{SWES}$, by first establishing a reflection between $\mathcal{WES}$ and $\mathcal{SWES}$.

For defining a map *ws* from W-event structures to stable W-event structures, it is not sufficient to simply add configurations to ensure that (W3) is satisfied, because this may destroy the condition (W2). This is illustrated in the following example.

Example 6.2.2

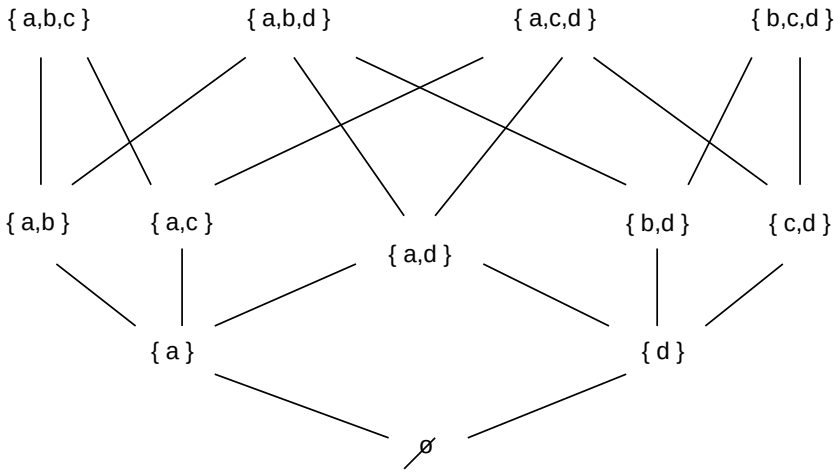


Figure 6.5: The non-stable W-event structure W_2

Let $W_2 = (E_2, C_2)$ be the non-stable W-event structure depicted in Figure 6.5. Since $\{a, b\} \uparrow \{b, d\}$ in this W-event structure, $\{b\}$ must be added to C_2 in order to extend C_2 to the set of configurations of a stable W-event structure. Similarly, $\{a, c\} \uparrow \{c, d\}$ implies that also $\{c\}$ must be added to C_2. The resulting set $C_2 \cup \{\{b\}, \{c\}\}$ satisfies (W3), but it does not satisfy (W2) anymore. Since $\{b\} \uparrow \{c\}$, we also have to add $\{b, c\}$. □

This now leads to the following definition of the map ws.

Given a W-event structure $W = (E, C)$, define $C^{(i)} \subseteq P_F(E)$ with $i \geq 0$ inductively by:

- $C^{(0)} = C$
- $\forall i \geq 1.\, C^{(i)} = C^{(i-1)} \cup \{c \cup c', c \cap c' \mid c, c' \in C^{(i-1)} \text{ with } c \uparrow c' \text{ in } C^{(i-1)}\}$.

Now define $ws(W) = (E, \hat{C})$ where $\hat{C} = \bigcup_{i \geq 0} C^{(i)}$.

Thus for the W-event structure $W_2 = (E_2, C_2)$ considered in Example 6.2.2, we have that $C_2^{(1)} = C_2 \cup \{\{b\}, \{c\}\}$, $C_2^{(2)} = C^{(1)} \cup \{\{b, c\}\}$, and $C_2^{(i)} = C_2^{(i-1)}$ for all $i \geq 3$. Hence $\hat{C}_2 = C_2 \cup \{\{b\}, \{c\}, \{b, c\}\}$.

Lemma 6.2.3

Let $W = (E, C)$ be a W-event structure. Then $ws(W) = (E, \hat{C})$ is a stable W-event structure.

Proof.

In order to prove that $ws(W)$ satisfies (W1), let $\emptyset \neq c \in \hat{C}$. Let $k \geq 0$ be minimal such that $c \in C^{(k)}$. We prove by induction on k that there exists $e \in c$ such that $c - e \in C^{(k)} \subseteq \hat{C}$. If $k = 0$ then $c \in C$ and since W satisfies (W1), there exists $e \in c$ such that $c - e \in C = C^{(0)}$. Now suppose that $k \geq 1$. Then by the minimality of k there exist $c_1, c_2 \in C^{(k-1)}$ with $c_1 \uparrow c_2$ such that $c = c_1 \cup c_2$ or $c = c_1 \cap c_2$. By the induction hypothesis there exist $e_1, \ldots, e_n \in E$ with $n = |c_1|$ such that $c_1 = \{e_1, \ldots, e_n\}$ and $\{e_1, \ldots, e_i\} \in C^{(k-1)}$ for all $0 \leq i \leq n$. By the minimality of k, $c_1 \neq c$ and $c_2 \neq c$.

First assume that $c = c_1 \cup c_2$. Let m be the largest integer such that $m \in \{1, \ldots, n\}$ and $e_m \notin c_2$. Hence $e_{m+1}, \ldots, e_n \in c_2$. Then, by the definition of $C^{(k)}$, $\{e_1, \ldots, e_{m-1}\} \cup c_2 = c - e_m \in C^{(k)}$.

Now assume that $c = c_1 \cap c_2$. Let m be the largest integer such that $m \in \{1, \ldots, n\}$ and $e_m \in c_2$. Hence $e_{m+1}, \ldots, e_n \notin c_2$. Then, by the definition of $C^{(k)}$, $\{e_1, \ldots, e_{m-1}\} \cap c_2 = c - e_m \in C^{(k)}$.

This proves that $ws(W)$ satisfies (W1). From the definition of $ws(W)$ we immediately have that $ws(W)$ satisfies (W2) and (W3). □

The map ws is now extended to morphisms by defining $ws(f) = f$ for each WES-morphism f.

Lemma 6.2.4

Let f be a WES-morphism from $W_1 = (E_1, C_1)$ to $W_2 = (E_2, C_2)$. Then $ws(f)$ is a WES-morphism from $ws(W_1) = (E_1, \hat{C}_1)$ to $ws(W_2) = (E_2, \hat{C}_2)$.

Proof.

Let $c \in \hat{C}_1$. It must be proved that $f(c) \in \hat{C}_2$ and that f is injective on c.

Let $k \geq 0$ be minimal such that $c \in C_1^{(k)}$. We prove by induction on k that $f(c) \in C_2^{(k)} \subseteq \hat{C}_2$ and that f is injective on c. If $k = 0$ then $c \in C_1$ and hence $f(c) \in C_2 = C_2^{(0)}$. Since f is a WES-morphism from W_1 to W_2, f is injective on c. Now assume that $k \geq 1$. Then there exist $c_0, c_1, c_2 \in C_1^{(k-1)}$ with $c_1 \subseteq c_0$ and $c_2 \subseteq c_0$ such that

$c = c_1 \cup c_2$ or $c = c_1 \cap c_2$. By the induction hypothesis $f(c_0), f(c_1), f(c_2) \in C_2^{(k-1)}$ and f is injective on c_0. Hence f is also injective on c. Now $f(c_1) \subseteq f(c_0)$ and $f(c_2) \subseteq f(c_0)$ and so by the definition of $C_2^{(k)}$ it follows that $f(c_1 \cup c_2) = f(c_1) \cup f(c_2) \in C_2^{(k)}$ and $f(c_1 \cap c_2) = f(c_1) \cap f(c_2) \in C_2^{(k)}$. This proves that $f(c) \in C_2^{(k)}$. □

Lemma 6.2.3 and Lemma 6.2.4 yield the following result.

Theorem 6.2.5

ws is a functor from $\mathcal{WES}$ to $\mathcal{SWES}$. □

As the next theorem shows *ws* is the left adjoint to the inclusion functor i from $\mathcal{SWES}$ to $\mathcal{WES}$. The co-unit of this adjunction is given by the identity arrows id_W for each stable W-event structure W. Hence the adjunction is a reflection.

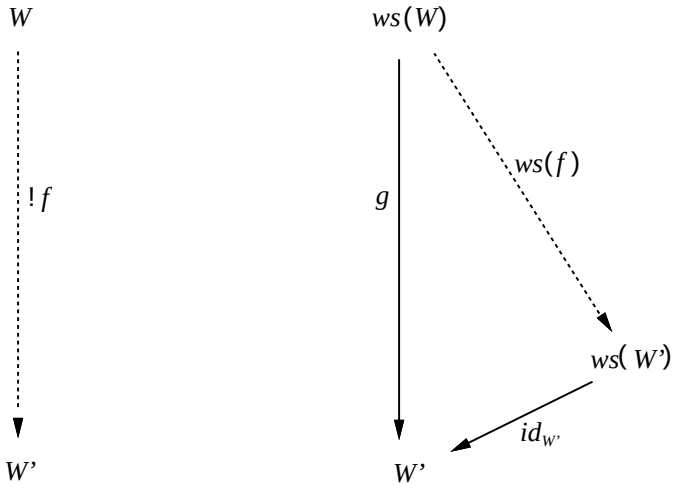
Theorem 6.2.6

$ws : \mathcal{WES} \to \mathcal{SWES}$ and $i : \mathcal{SWES} \to \mathcal{WES}$ form a reflection with *ws* the left adjoint and the identity arrows id_W as co-unit.

Proof.

Note that the co-unit is well-defined because $ws(W) = W$ for each stable W-event structure W.

Let $W = (E, C)$ be a W-event structure, let $W' = (E', C')$ be a stable W-event structure, and let g be a WES-morphism from $ws(W) = (E, \hat{C})$ to W'. Then we must prove that there exists a unique WES-morphism f from W to W' such that the following diagram commutes.



Since *ws* is the identity on arrows, it is sufficient to prove that g is a WES-morphism from W to W'. This however follows immediately from the observation that $C \subseteq \hat{C}$. □

The reflections from Theorem 6.1.9 and Theorem 6.2.6 can now be composed which yields the following result.

Theorem 6.2.7
$ws \circ ew : \mathcal{LES} \to \mathcal{SWES}$ and $we \circ i : \mathcal{SWES} \to \mathcal{LES}$ form a reflection with $ws \circ ew$ the left adjoint and the identity arrows id_W as co-unit. □

In Corollary 6.1.10 a characterization is given of the L-event structures representing W-event structures, by cutting the reflection between $\mathcal{LES}$ and $\mathcal{WES}$ down to an equivalence. We now give a similar result for stable W-event structures.

In order to identify a full subcategory for which the reflection in Theorem 6.2.7 cuts down to an equivalence, define the following "backward closure" property for arbitrary L-event structures $(E, C, \vdash)$.

(BC) $c \uparrow c' \Rightarrow c \cap c' \in C$.

Then the following result follows easily from Theorem 6.2.7.

Corollary 6.2.8
$\mathcal{SWES}$ is equivalent to the full subcategory of $\mathcal{LES}$ the objects of which satisfy (FC), (D), and (BC). □

6.3 UL-Event Structures and G-Prime Event Structures

In this section we investigate the relationship between UL-event structures and the generalization of prime event structures from [96]. These event structures from [96] will be referred to as *G-prime event structures* in what follows. We prove that, similar to the relationship between the category of L-event structures and the category of (stable) W-event structures, there exists a reflection between the category of UL-event structures and the category of G-prime event structures, with the left adjoint going from UL-event structures to G-prime event structures.

A G-prime event structure has a global *consistency predicate* which specifies which (finite) sets of events are *not* in conflict. A consequence is that, in contrast to the binary conflict relation of prime event structures, G-prime event structures can have arbitrary sets of conflicting events. In particular a set of events can be in conflict, while each of its non-trivial finite subsets is conflict-free.

Definition 6.3.1
$\mathcal{GPES}$ is the category of *G-prime event structures* specified as follows.
An object of $\mathcal{GPES}$ is a G-prime event structure $G = (E, Con, \leq)$ where E is a set of events, $\leq \subseteq E \times E$ is a partial order, the causal dependency relation, and $Con \subseteq P_F(E)$ is the *consistency predicate* such that

(G0) $\forall e \in E.\, \{e\} \in Con$

(G1) $\forall e \in E.\ \downarrow e$ is finite

(G2) $\forall X, Y \subseteq E.(Y \subseteq X \in Con \Rightarrow Y \in Con)$

(G3) $\forall X \in Con.\forall e, e' \in E.(e' \leq e \in X \Rightarrow X \cup e' \in Con)$. □

An arrow of $\mathcal{GPES}$ is a *GPES-morphism* $f : (E_1, Con_1, \leq_1) \rightarrow (E_2, Con_1, \leq_2)$ which is a partial function $f : E_1 \rightarrow E_2$ such that

(1) $\forall e \in E_1.(f(e)$ is defined $\Rightarrow \downarrow f(e) \subseteq f(\downarrow e))$

(2) $\forall X \in Con_1.f(X) \in Con_2$

(3) $\forall X \in Con_1.\forall e_1, e_2 \in X.((f(e_1)$ and $f(e_2)$ are defined and $f(e_1) = f(e_2)) \Rightarrow e_1 = e_2)$. □

The identity morphism associated with an object is the identity function on its events and composition of GPES-morphisms is composition of partial functions. □

Let $G = (E, Con, \leq)$ be a G-prime event structure and let $c \subseteq E$. Again we say that c is *downward-closed* iff

$$\forall e, e' \in E.((e \in c \text{ and } e' \leq e) \Rightarrow e' \in c).$$

We say that c is *consistent* iff

$$\forall Y \in P_F(c).Y \in Con.$$

A *configuration* of G is a subset of events which is downward-closed and consistent. The set of all configurations of G is denoted by C_G. The set $C_G \cap P_F(E)$ of all *finite* configurations of G is denoted by FC_G.

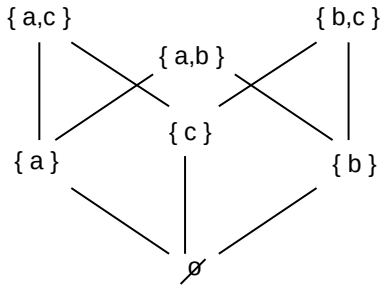


Figure 6.6: A G-prime event structure

In Figure 6.6 the configurations of a G-prime event structure are depicted. For this G-prime event structure the set $\{a, b, c\}$ is downward-closed, but not consistent. Since the elements in $\{a, b, c\}$ are however pairwise consistent, there is no ordinary prime event structure which has the same set of configurations.

The following lemma states some useful properties of G-prime event structures.

Lemma 6.3.2

Let $G = (E, Con, \leq)$ be a G-prime event structure. Then

(1) $\forall e \in E.\ \downarrow e \in FC_G$

(2) $\forall Y \subseteq E.(Y \in Con \Leftrightarrow \exists c \in C_G.Y \in P_F(c))$.

Proof.

(1) Let $e \in E$. Clearly, $\downarrow e$ is downward-closed. By (G1), $\downarrow e$ is finite. By (G2) it is now sufficient to prove that $\downarrow e \in Con$. By (G0), $\{e\} \in Con$. Hence repeatedly applying (G3) yields that $\downarrow e \in Con$.

(2) Suppose $Y \subseteq E$ is such that $Y \in Con$. Since Y is finite it is sufficient to prove that $\downarrow Y \in C_G$ and is finite. By (G1) and the finiteness of Y, $\downarrow Y$ is also finite. Repeatedly applying (G3), starting with Y, now yields that $\downarrow Y \in Con$. Since $\downarrow Y$ is downward-closed, we can conclude that $\downarrow Y \in C_G$. The converse implication follows immediately from the definition of C_G. □

Part (1) implies that, similar to the situation for prime event structures, there exists for each event of a G-prime event structure a minimal (finite) configuration containing this event. Part (2) states that the consistency predicate of a G-prime event structure can be derived from its set of configurations.

A characterization of GPES-morphisms in terms of configurations is stated in the next lemma.

Lemma 6.3.3

Let $G_1 = (E_1, Con_1, \leq_1)$ and $G_2 = (E_2, Con_2, \leq_2)$ be G-prime event structures and let $f : E_1 \to E_2$ be a partial function. Then f is a GPES-morphism iff

(1') $\forall c \in C_{G_1}.f(c) \in C_{G_2}$

(2') $\forall c \in C_{G_1}.\forall e_1, e_2 \in c.((f(e_1)$ and $f(e_2)$ are defined and $f(e_1) = f(e_2)) \Rightarrow e_1 = e_2)$.

Proof.

Suppose that f is a GPES-morphism. First we prove condition (2'). Let $c \in C_{G_1}$ and let $e_1, e_2 \in c$ be such that $f(e_1)$ and $f(e_2)$ are defined and $e_1 \neq e_2$. Then $\{e_1, e_2\} \subseteq c$ and hence $\{e_1, e_2\} \in Con_1$. This implies that $f(e_1) \neq f(e_2)$ by condition (3) in the definition of GPES-morphism. This proves condition (2').

Now in order to prove (1'), let $c \in C_{G_1}$. It must be proved that $f(c) \in C_{G_2}$. In order to prove that $f(c)$ is consistent, let $X \in P_F(f(c))$. We must prove that $X \in Con_2$. Let $Y = \{e \in c \mid f(e)$ is defined and $f(e) \in X\}$. Hence $X = f(Y)$. Because X is finite, we have that $Y \in P_F(c)$ by (2'). This implies that $Y \in Con_1$ because $c \in C_{G_1}$. We can now conclude that $X = f(Y) \in Con_2$ by condition (2) in the definition of GPES-morphism. Now in order to prove that $f(c)$ is downward-closed, suppose that $e_1 \in c$ and $e_2 \in E_2$ are such that $e_2 \leq_2 f(e_1) \in f(c)$. Then by condition (1) in the

definition of GPES-morphism $e_2 \in \downarrow f(e_1) \subseteq f(\downarrow e_1) \subseteq f(c)$. This proves that $f(c)$ is downward-closed, and hence that $f(c) \in C_{G_2}$.

Now suppose that f satisfies the conditions (1') and (2') above. In order to prove condition (1) in the definition of GPES-morphism, let $e_1 \in E_1$ be such that $f(e_1)$ is defined and suppose $e_2 \in E_2$ is such that $e_2 \in \downarrow f(e_1)$, that is $e_2 \leq_2 f(e_1)$. Then by Lemma 6.3.2(1), $\downarrow e_1 \in C_{G_1}$ and hence also $f(\downarrow e_1) \in C_{G_2}$ by condition (1'). This implies that $f(\downarrow e_1)$ is downward-closed, and hence that $e_2 \in f(\downarrow e_1)$.

In order to prove condition (2) in the definition of GPES-morphism, suppose $X \in Con_1$. Then by Lemma 6.3.2(2) there exists $c \in C_{G_1}$ such that $X \in P_F(c)$. By condition (1') we then have that $f(c) \in C_{G_2}$. Because $f(X) \in P_F(f(c))$ this implies that $f(X) \in Con_2$.

Finally, condition (3) in the definition of GPES-morphism follows immediately from Lemma 6.3.2(2) and condition (2'). □

Using its finite configurations, we now demonstrate that each G-prime event structure can be viewed as an UL-event structure. This is done by interpreting again the diamonds in the configuration structure of a G-prime event structure as concurrency expressed through the enabling relation.

Let $G = (E, Con, \leq)$ be a G-prime event structure. Then define $gu(G) = (E, FC_G, \vdash)$ where $\vdash \subseteq FC_G \times P_F(E)$ is given by:

$$c \vdash u \Leftrightarrow c \cap u = \emptyset \text{ and } \forall v \subseteq u.\, c \cup v \in FC_G.$$

Lemma 6.3.4

Let $G = (E, Con, \leq)$ be a G-prime event structure. Then $gu(G) = (E, FC_G, \vdash)$ is an L-event structure.

Proof.

In order to prove that $gu(G)$ satisfies (E0), let $\emptyset \neq c \in FC_G$. Let $e \in c$ be a maximal event in c in the sense that for all $e' \in c$, $e \leq e'$ implies that $e = e'$. Then $c - e \in FC_G$ and hence $c - e \vdash e$. This proves that $gu(G)$ satisfies (E0). From the definition of $gu(G)$ it easily follows that $gu(G)$ satisfies (E1) and (E2). □

Our next aim is to prove that for each G-prime event structure G, the L-event structure $gu(G)$ has the unique occurrence property. The first step is to show that two step firing sequences of $gu(G)$ that lead to the same configuration have the same past (under $\approx_{gu(G)}$).

Lemma 6.3.5

Let $G = (E, Con, \leq)$ be a G-prime event structure with $gu(G) = (E, FC_G, \vdash)$ and let $\rho_1, \rho_2 \in SFS_{gu(G)}$ be such that $alph(\rho_1) = alph(\rho_2)$. Then $past(\rho_1) = past(\rho_2)$.

Proof.

The proof is by induction on $k = |\rho_1|$. If $k = 0$ then $\rho_1 = \rho_2 = \emptyset$ and the claim clearly holds. Now assume that $k > 0$. Then there exist $\rho_1', \rho_2' \in SFS$ and $\emptyset \neq u_1, u_2 \in P_F(E)$ such that $\rho_1 = \rho_1' u_1$, $\rho_2 = \rho_2' u_2$, $cf(\rho_1') \vdash u_1$, and $cf(\rho_2') \vdash u_2$. Let $e_1 \in u_1$ and $e_2 \in u_2$. Since $gu(G)$ satisfies (E2), $\rho_1'(u_1 - e_1)e_1, \rho_2'(u_2 - e_2)e_2 \in SFS$.

Moreover, because $\approx_{gu(G)}$ satisfies (C1), $past(\rho_1) = past(\rho_1'(u_1 - e_1)e_1)$ and $past(\rho_2) = past(\rho_2'(u_2 - e_2)e_2)$.

If $e_1 = e_2$ then $alph(\rho_1'(u_1 - e_1)) = alph(\rho_2'(u_2 - e_2))$ and hence by the induction hypothesis $past(\rho_1'(u_1 - e_1)) = past(\rho_2'(u_2 - e_2))$. This implies, because $\approx_{gu(G)}$ satisfies (C2), that $\rho_1'(u_1 - e_1)e_1 \approx_{gu(G)} \rho_2'(u_2 - e_2)e_2$. Thus $past(\rho_1) = past(\rho_1'(u_1 - e_1)e_1) = past(\rho_1'(u_1 - e_1)) \cup \langle \rho_1'(u_1 - e_1)e_1 \rangle = past(\rho_2'(u_2 - e_2)) \cup \langle \rho_2'(u_2 - e_2)e_2 \rangle = past(\rho_2'(u_2 - e_2)e_2) = past(\rho_2)$.

Now assume that $e_1 \neq e_2$. Then it is easy to see that $alph(\rho_1) - \{e_1, e_2\} \in FC_G$. By Lemma 6.3.4 and Lemma 5.2.5(2), there exists $\rho \in SFS$ such that $alph(\rho) = alph(\rho_1) - \{e_1, e_2\}$. Since $\rho e_1 \in SFS$ and $alph(\rho e_1) = alph(\rho_2'(u_2 - e_2))$, we have by the induction hypothesis that $past(\rho e_1) = past(\rho_2'(u_2 - e_2))$. Similarly, $past(\rho e_2) = past(\rho_1'(u_1 - e_1))$. Hence, because $\approx_{gu(G)}$ satisfies (C2), $\rho e_1 e_2 \approx_{gu(G)} \rho_2'(u_2 - e_2)e_2$ and $\rho e_2 e_1 \approx_{gu(G)} \rho_1'(u_1 - e_1)e_1$. Since $alph(\rho) \vdash \{e_1, e_2\}$ we also have that $\rho e_1 \approx_{gu(G)} \rho e_2 e_1$ and $\rho e_2 \approx_{gu(G)} \rho e_1 e_2$. Summarizing these results we can conclude that $past(\rho_1) = past(\rho_1'(u_1 - e_1)e_1) = past(\rho_1'(u_1 - e_1)) \cup \langle \rho_1'(u_1 - e_1)e_1 \rangle = past(\rho e_2) \cup \langle \rho e_2 e_1 \rangle = past(\rho) \cup \langle \rho e_2 \rangle \cup \langle \rho e_2 e_1 \rangle = past(\rho) \cup \langle \rho e_1 e_2 \rangle \cup \langle \rho e_1 \rangle = past(\rho e_1) \cup \langle \rho e_1 e_2 \rangle = past(\rho_2'(u_2 - e_2)) \cup \langle \rho_2'(u_2 - e_2)e_2 \rangle = past(\rho_2'(u_2 - e_2)e_2) = past(\rho_2)$. □

Lemma 6.3.6

Let $G = (E, Con, \leq)$ be a G-prime event structure. Then $gu(G) = (E, FC_G, \vdash)$ is an UL-event structure.

Proof.

By Lemma 6.3.4, $gu(G)$ is an L-event structure. We must show that $gu(G)$ has the unique occurrence property as stated in Definition 5.3.7.

Let $e \in E$. Then $\downarrow e - e, \downarrow e \in FC_G$ by Lemma 6.3.2(1) and hence $\downarrow e - e \vdash e$. By Lemma 6.3.4 and Lemma 5.2.5(2), there exists $\rho \in SFS$ such that $alph(\rho) = \downarrow e - e$.

Then $\rho e \in PI$ and hence condition (U1) is satisfied. In order to prove that condition (U2) is satisfied, we first show that $\rho e \approx_{gu(G)} \rho' e$ for all $\rho' e \in PI$. Then by the transitivity of $\approx_{gu(G)}$ we have that also $\rho' e \approx_{gu(G)} \rho'' e$ for all $\rho' e, \rho'' e \in PI$.

So let $\rho' e \in PI$. Then $alph(\rho' e) \in FC_G$ and hence $alph(\rho) \subseteq alph(\rho')$. We prove that $\rho e \approx_{gu(G)} \rho' e$ by induction on $|\rho'|$. If $alph(\rho') = alph(\rho)$ then $past(\rho) = past(\rho')$ by Lemma 6.3.5. Hence, because $\approx_{gu(G)}$ satisfies (C2), $\rho e \approx_{gu(G)} \rho' e$. Now assume that $|\rho'| > |\rho|$. Then there exists $e' \in alph(\rho') - alph(\rho)$ such that e' is a maximal element in $alph(\rho')$ under $<$. Such an e' must exist because $alph(\rho')$ is a finite set and $<$ is a partial ordering relation. Then $alph(\rho') - e' \in FC_G$ and $(alph(\rho') - e') \cup e \in FC_G$. Let $\rho'' \in SFS$ be such that $alph(\rho'') = alph(\rho') - e'$. Then $\rho'' e \in PI$. Since $|\rho''| < |\rho'|$, $\rho'' e \approx_{gu(G)} \rho e$ by the induction hypothesis. Now $alph(\rho'' e') = alph(\rho')$ and hence by Lemma 6.3.5, $past(\rho'' e') = past(\rho')$. Then because $\approx_{gu(G)}$ satisfies (C2), $\rho'' e' e \approx_{gu(G)} \rho' e$. Since $alph(\rho'') \vdash \{e, e'\}$ and $\approx_{gu(G)}$ satisfies (C1), we also have that $\rho'' e' e \approx_{gu(G)} \rho'' e$. We can now conclude that $\rho e \approx_{gu(G)} \rho'' e \approx_{gu(G)} \rho'' e' e \approx_{gu(G)} \rho' e$. This proves condition (U2). □

As to be expected, not every UL-event structure can be obtained from a G-prime event structure by applying gu. For instance, the UL-event structure ES_3 in Example 5.2.2 cannot be the UL-event structure associated with any G-prime event structure.

In order to extend gu to a functor, define $gu(f) = f$ for each GPES-morphism f.

Lemma 6.3.7

Let f be a GPES-morphism from $G_1 = (E_1, Con_1, \leq_1)$ to $G_2 = (E_2, Con_2, \leq_2)$. Then $gu(f)$ is an LES-morphism from $gu(G_1) = (E_1, FC_{G_1}, \vdash_1)$ to $gu(G_2) = (E_2, FC_{G_2}, \vdash_2)$.

Proof.

Suppose that $c \vdash_1 u$. Then $c \cap u = \emptyset$ and $c \cup u \in FC_{G_1}$. So by condition (2') in Lemma 6.3.3, $f(c) \cap f(u) = \emptyset$. We also have that $c \cup v \in FC_{G_1}$ for all $v \subseteq u$. Thus by condition (1') in Lemma 6.3.3, $f(c \cup v) = f(c) \cup f(v) \in FC_{G_2}$ for all $v \subseteq u$. Consequently, $f(c) \vdash_2 f(u)$. □

The following result now follows immediately from Lemma 6.3.6 and Lemma 6.3.7.

Theorem 6.3.8

gu is a functor from $\mathcal{GPES}$ to $\mathcal{ULES}$. □

Next we define a functor from $\mathcal{ULES}$ to $\mathcal{GPES}$. This is done by first associating with each L-event structure a structure $ug(ES)$. In $ug(ES)$ a set of events is consistent iff the events occur together in some configuration. An event causally depends on another event iff its occurrence implies that this other event has already occurred.

So let $ES = (E, C, \vdash)$ be an L-event event structure. Then define $ug(ES) = (E, Con, \leq)$ where

- $\leq \subseteq E \times E$ is given by $e_1 \leq e_2 \Leftrightarrow \forall c \in C.\, (e_2 \in c \Rightarrow e_1 \in c)$
- $Con \subseteq P_F(E)$ is given by $Con = \{X \in P_F(E) \mid \exists c \in C.\, X \subseteq c\}$.

If in ES all events occur in a configuration, then we can prove that $ug(ES)$ is a G-prime event structure.

Lemma 6.3.9

Let $ES = (E, C, \vdash)$ be an L-event structure which satisfies condition (U1) in the definition of the unique occurrence property. Then $ug(ES) = (E, Con, \leq)$ is a G-prime event structure.

Proof.

Clearly, $\leq$ is reflexive and transitive. In order to prove that $\leq$ is anti-symmetric, suppose $e_1, e_2 \in E$ are such that $e_1 \leq e_2$ and $e_2 \leq e_1$. Then, for all $c \in C$, $e_1 \in c$ iff $e_2 \in c$. By (U1) there exists $c \in C$ such that $e_1 \in c$. By Lemma 5.2.3 there is a c' such that $e_1 \neq e_2$ implies $e_1 \in c'$ iff $e_2 \notin c'$. Hence $e_1 = e_2$. This proves that $\leq$ is a partial order.

In order to prove that $ug(ES)$ satisfies (G0) and(G1), let $e \in E$. Then by (U1), there exists $c \in C$ such that $e \in c$. Hence $\{e\} \in Con$. Moreover, $\downarrow e \subseteq c$ and hence $\downarrow e$ is finite. This proves that $ug(ES)$ satisfies conditions (G0) and (G1). Conditions (G2) and (G3) follow easily from the definition of $ug(ES)$. □

As usual ug is to be extended to a functor by defining $ug(f) = f$ for each LES-morphism f. In proving that $ug(f)$ is a GPES-morphism, condition (U2) in the definition of the unique occurrence property is essential. This is illustrated in the following example.

Example 6.3.10

Let ES_{12} and ES_{13} be the L-event structures depicted in Figure 6.7. Note that ES_{12} is not an UL-event structure.

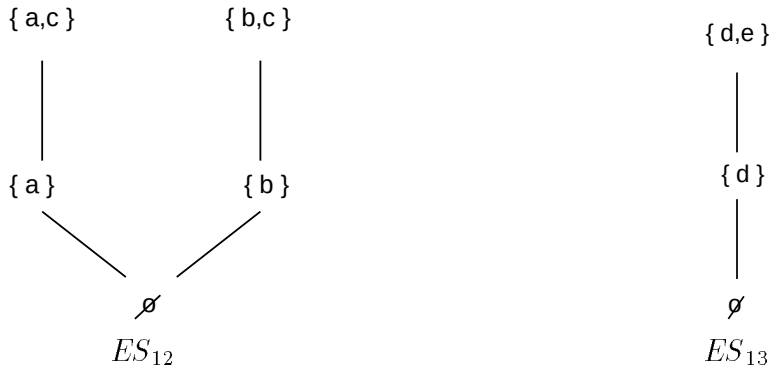


Figure 6.7: L-event structures ES_{12} and ES_{13}

Define f by $f(a) = f(b) = d$ and $f(c) = e$. Then f is an LES-morphism from ES_{12} to ES_{13}. Since $\{c\} \in C_{ug(ES_{12})}$ while $f(\{c\}) = \{e\} \notin C_{ug(ES_{13})}$, Lemma 6.3.3 implies that $ug(f)$ is not a GPES-morphism from $ug(ES_{12})$ to $ug(ES_{13})$. □

Hence arbitrary LES-morphisms are not preserved under ug. LES-morphisms between L-event structures with the unique occurrence property are however preserved under ug.

Lemma 6.3.11

Let f be an LES-morphism from $ES_1 = (E_1, C_1, \vdash_1)$ to $ES_2 = (E_2, C_2, \vdash_2)$ where ES_1 and ES_2 are UL-event structures. Then $ug(f)$ is a GPES-morphism from $ug(ES_1) = (E_1, Con_1, \leq_1)$ to $ug(ES_2) = (E_2, Con_2, \leq_2)$.

Proof.

In order to prove condition (1) in the definition of GPES-morphism, let $e \in E_1$ be such that $f(e)$ is defined and suppose $e' \in {\downarrow} f(e)$. We prove that $e' \in f({\downarrow} e)$. If $e' = f(e)$ then we are done, so assume that $e' \neq f(e)$. Let $\rho \in SFS_{ES_1}$ be such that $\rho e \in PI_{ES_1}$. By condition (U1) in the definition of the unique occurrence property such ρ exists. Then $alph(\rho e) \in C_1$. Hence, because f is an LES-morphism, $f(alph(\rho e)) \in C_2$. Since $f(e) \in f(alph(\rho e))$ this implies, because $e' \leq_2 f(e)$ and $e' \neq f(e)$, that $e' \in f(alph(\rho))$. Let $e'' \in alph(\rho)$ be such that $f(e'') = e'$. If $e'' \leq_1 e$, then $e' = f(e'') \in f({\downarrow} e)$.

In order to prove that $e'' \leq_1 e$, define $R \subseteq PI_{ES_1} \times PI_{ES_1}$ by: $\rho_1 e_1 \, R \, \rho_2 e_2$ iff $(e_1 = e_2 \neq e$ or $(e_1 = e_2 = e$ and $(e'' \in alph(\rho_1) \Leftrightarrow e'' \in alph(\rho_2))))$. Assume that R is an equivalence relation which is SFS_{ES_1}-consistent. Then $\approx_{ES_1} \subseteq R$ because $\approx_{ES_1}$ is the least equivalence relation which is SFS_{ES_1}-consistent. Since $\rho e \in PI_{ES_1}$, $e'' \in alph(\rho)$, and ES_1 has the unique occurrence property it then follows that for all $\rho_1 e \in PI_{ES_1}$, $e'' \in alph(\rho_1)$. Hence $e'' \in c$ for all $c \in C_1$ such that $e \in c$. Thus $e'' \leq_1 e$.

Consequently, what remains to be proved is that R is an equivalence relation which satisfies (C1) and (C2).

Clearly, R is an equivalence relation. In order to prove that R satisfies (C1), suppose $\rho_1 u \in SFS_{ES_1}$ and $e_1 \in u$. If $e_1 \neq e$ then it is clear that $\rho_1 e_1 \, R \, \rho_1 (u - e_1) e_1$, so assume that $e_1 = e$. If $e'' \notin u$ then it is clear that $\rho_1 e_1 \, R \, \rho_1 (u - e_1) e_1$. We now show that $e'' \in u$ leads to a contradiction. To see this, suppose that $e'' \in u$. Since $alph(\rho_1 e_1) \in C_1$ and f is an LES-morphism, we must have that $f(alph(\rho_1 e_1)) = alph(f(\rho_1)) \cup f(e) \in C_2$. Combining this with $e' \leq_2 f(e)$ and $e' \neq f(e)$ yields that $e' \in alph(f(\rho_1))$. On the other hand, we also have that $alph(\rho_1) \vdash_1 e''$ and hence by the definition of LES-morphism $f(alph(\rho_1)) \vdash_2 f(e'')$. This leads to a contradition with $f(e'') = e' \in alph(f(\rho_1)) = f(alph(\rho_1))$. We can now conclude that $e'' \in u$ is not possible. This proves that R satisfies (C1).

Now in order to prove that R satisfies (C2), let $\rho_1 e_1, \rho_2 e_1 \in PI_{ES_1}$ be such that $past_R(\rho_1) = past_R(\rho_2)$. If $e_1 \neq e$ then we immediately have that $\rho_1 e_1 \, R \, \rho_2 e_1$. If $e_1 = e$, then $\rho_1 e_1 \, R \, \rho_2 e_1$ because $past_R(\rho_1) = past_R(\rho_2)$ implies that also $alph(\rho_1) = alph(\rho_2)$. This proves that R satisfies (C2).

Thus R is an equivalence relation satisfying (C1) and (C2) which completes the proof that $ug(f)$ satisfies condition (1) in the definition of GPES-morphism.

In order to prove condition (2) in the definition of GPES-morphism, let $X \in Con_1$. Then $X \subseteq c$ for some $c \in C_1$. Since f is an LES-morphism, this implies that $f(X) \subseteq f(c) \in C_2$. Hence $f(X) \in Con_2$.

Finally, condition (3) in the definition of GPES-morphism follows immediately from Lemma 5.2.7. □

The following result is an immediate consequence of Lemma 6.3.9 and Lemma 6.3.11.

Theorem 6.3.12
ug is a functor from $\mathcal{ULES}$ to $\mathcal{GPES}$. □

Now we prove that ug and gu form an adjunction.

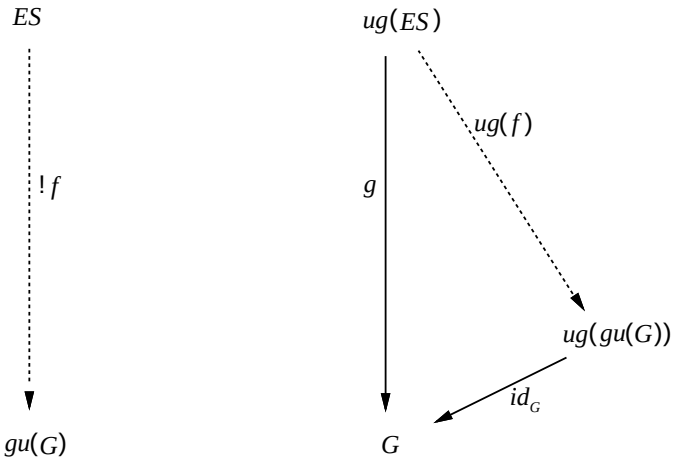
Theorem 6.3.13
$ug : \mathcal{ULES} \to \mathcal{GPES}$ and $gu : \mathcal{GPES} \to \mathcal{ULES}$ form a reflection with ug the left adjoint and the identity arrows id_G as co-unit.

Proof.
Let $G = (E, Con, \leq)$ be a G-prime event structure with $gu(G) = (E, FC_G, \vdash)$ and $ug(gu(G)) = (E, Con', \leq')$. First we prove that $G = ug(gu(G))$.

If $X \in Con$, then it is easy to see that $\downarrow X \in FC_G$, and hence $X \in Con'$. Conversely, $X \in Con'$ implies that there exists $c \in FC_G$ such that $X \subseteq c$, and hence $X \in Con$. This proves that $Con = Con'$.

If $e_1, e_2 \in E$ are such that $e_1 \leq e_2$, then $e_1 \in c$ for all $c \in FC_G$ such that $e_2 \in c$, and hence also $e_1 \leq' e_2$. Conversely, if $e_1 \leq' e_2$ then we use the observation that $\downarrow e_2 \in FC_G$ by Lemma 6.3.2(1) to conclude that $e_1 \in \downarrow e_2$. This proves that $G = ug(gu(G))$, and hence that id_G is a GPES-isomorphism from $ug(gu(G))$ to G.

Now let $ES = (E'', C, \vdash'')$ be an UL-event structure and let g be a GPES-morphism from $ug(ES) = (E'', Con'', \leq'')$ to G. Then we must prove that there exists a unique LES-morphism f from ES to $gu(G)$ such that the following diagram commutes.



Since ug is the identity on arrows, it is sufficient to prove that g is an LES-morphism from ES to $gu(G)$. Suppose $c \vdash'' u$. Then $c \cap u = \emptyset$ and $c \cup v \in C \subseteq FC_{ug(ES)}$, for all $v \subseteq u$, by (E2). Since g is a GPES-morphism from $ug(ES)$ to G we now have by Lemma 6.3.3 that $g(c) \cup g(v) \in FC_G$ for all $v \subseteq u$ and $g(c) \cap g(u) = \emptyset$. Hence $g(c) \vdash g(u)$. □

The conditions (FC), (BC), and (D) have been shown in Section 6.2 to characterize within the category $\mathcal{LES}$ a full subcategory equivalent to $\mathcal{SWES}$, the category of stable W-event structures. If we restrict this subcategory further to the full subcategory of UL-event structures satisfying (FC), (BC), and (D), we obtain a category equivalent to $\mathcal{GPES}$.

In order to prove this, we first need the following lemma.

Lemma 6.3.14

Let $ES = (E, C, \vdash)$ be an UL-event structure which satisfies (BC) with $ug(ES) = (E, Con, \leq)$ and let $e \in E$. Then $\downarrow e \in C$.

Proof.

Define $R \subseteq PI_{ES} \times PI_{ES}$ by: $\rho_1 e_1 \, R \, \rho_2 e_2$ iff ($e_1 = e_2 \neq e$ or ($e_1 = e_2 = e$ and $\exists c \in C.(c \subseteq alph(\rho_1) \cap alph(\rho_2)$ and $c \cup e \in C)$)). We first prove that R is an equivalence relation satisfying (C1) and (C2).

Clearly, R is reflexive and symmetric. In order to prove that R is transitive, suppose that $\rho_1 e_1 \, R \, \rho_2 e_2 \, R \, \rho_3 e_3$. The only non-trivial case is that $e_1 = e_2 = e_3 = e$. Then there exists $c_1 \in C$ such that $c_1 \subseteq alph(\rho_1) \cap alph(\rho_2)$ and $c_1 \cup e \in C$ and there exists $c_2 \in C$ such that $c_2 \subseteq alph(\rho_2) \cap alph(\rho_3)$ and $c_2 \cup e \in C$. Then $c_1 \uparrow c_2$ and $(c_1 \cup e) \uparrow (c_2 \cup e)$

and hence by (BC), $c_1 \cap c_2 \in C$ and $(c_1 \cup e) \cap (c_2 \cup e) = (c_1 \cap c_2) \cup e \in C$. Because $(c_1 \cap c_2) \subseteq alph(\rho_1) \cap alph(\rho_3)$ this proves that $\rho_1 e_1 \, R \, \rho_3 e_3$. We can now conclude that R is transitive, and hence that R is an equivalence relation.

In order to prove that R satisfies (C1), let $\rho \in SFS_{ES}$ and $u \in P_F(E)$ be such that $\rho u \in SFS_{ES}$ and let $e' \in u$. If $e' \neq e$ then it is clear that $\rho e' \, R \, \rho(u - e')e'$, so assume that $e' = e$. Then $alph(\rho) \in C$ is such that $alph(\rho) \subseteq alph(\rho) \cap alph(\rho(u - e'))$ and $alph(\rho) \cup e \in C$. Hence $\rho e' \, R \, \rho(u - e')e'$. This proves that R satisfies (C1).

In order to prove that R satisfies (C2), let $\rho_1 e', \rho_2 e' \in PI_{ES}$ be such that $past_R(\rho_1) = past_R(\rho_2)$. If $e' \neq e$ then it is clear that $\rho_1 e' \, R \, \rho_2 e'$. If $e = e'$, then $past_R(\rho_1) = past_R(\rho_2)$ implies that also $alph(\rho_1) = alph(\rho_2)$. Hence $alph(\rho_1) \in C$ is such that $alph(\rho_1) \subseteq alph(\rho_1) \cap alph(\rho_2)$ and $alph(\rho_1) \cup e \in C$. This proves that $\rho_1 e' \, R \, \rho_2 e'$ and hence that R satisfies (C2).

Because $\approx_{ES}$ is the least equivalence relation satisfying (C1) and (C2), we must have that $\approx_{ES} \subseteq R$. Now let $c \in C$ be such that $e \in c$ and c is minimal in the sense that $\forall c' \in C.((e \in c' \text{ and } c' \subseteq c) \Rightarrow c = c')$. Note that c exists by condition (U1) in the definition of the unique occurrence property. Then by the minimality of c and (E0) we have that $c - e \vdash e$. We also have that $\downarrow e \subseteq c$ by the definition of $\leq$. Now assume to the contrary that $\downarrow e \neq c$. Then there exists $e' \in c$ such that $e' \not\leq e$. Hence there exists $c' \in C$ such that $e \in c'$ and $e' \notin c'$. We may assume that c' is minimal in the sense that $\forall c'' \in C.((e \in c'' \text{ and } e' \notin c'' \text{ and } c'' \subseteq c') \Rightarrow c'' = c')$. Then $c' - e \vdash e$ by the minimality of c' and (E0). Now let $\rho, \rho' \in SFS_{ES}$ be such that $alph(\rho) = c - e$ and $alph(\rho') = c' - e$. Hence $\rho e, \rho' e \in PI_{ES}$. By condition (U2) in the definition of the unique occurrence property, $\rho e \approx_{ES} \rho' e$, and hence $\rho e \, R \, \rho' e$ because $\approx_{ES} \subseteq R$. This implies that there exists $c'' \in C$ such that $c'' \subseteq alph(\rho) \cap alph(\rho')$ and $c'' \cup e \in C$. Then $c'' \cup e \subseteq c$ and $e \in c'' \cup e$, but $c'' \cup e \neq c$ because $e' \in c - (c'' \cup e)$, a contradiction with the minimality of c. We can now conclude that $\downarrow e = c \in C$. □

Lemma 6.3.15

Let ES be an UL-event structure which satisfies (FC), (BC), and (D). Then $ES = gu(ug(ES))$.

Proof.

Let $ES = (E, C, \vdash)$, $ug(ES) = (E, Con, \leq)$, and $gu(ug(ES)) = (E, FC_{ug(ES)}, \vdash')$. First we prove that $FC_{ug(ES)} = C$.

Clearly, $C \subseteq FC_{ug(ES)}$. In order to prove that $FC_{ug(ES)} \subseteq C$, let $c \in FC_{ug(ES)}$. Then $c \in Con$, and hence there exists $c' \in C$ such that $c \subseteq c'$. By Lemma 6.3.14, $\downarrow e \in C$ for all $e \in c$. Because $c = \bigcup\{\downarrow e \mid e \in c\}$, we then have that $c \in C$ by repeatedly applying (FC). This proves that $FC_{ug(ES)} \subseteq C$, and hence that $C = FC_{ug(ES)}$.

In order to prove that $\vdash = \vdash'$, first assume that $c \vdash u$. Then $c \cap u = \emptyset$ and $c \cup v \in C = FC_{ug(ES)}$ for all $v \subseteq u$. Hence $c \vdash' u$ by the definition of $\vdash'$. Now assume that $c \vdash' u$. Then $c \cap u = \emptyset$ and $c \cup v \in FC_{ug(ES)} = C$ for all $v \subseteq u$. Hence $c \vdash u$ by condition (D). □

From the definition of gu it easily follows that for every G-prime event structure G, $gu(G)$ satisfies (FC), (BC), and (D). Hence we have the following result from Lemma 6.3.15 and Theorem 6.3.13.

Theorem 6.3.16
$\mathcal{GPES}$ is equivalent to the full subcategory of $\mathcal{ULES}$ the objects of which satisfy (FC), (D), and (BC). □

6.4 UL-Event Structures and Prime Event Structures

As the last point in our comparison of the various categories of event structures we want to prove that there exists also a reflection between the category of prime event structures and the category of UL-event structures. In view of Theorem 6.3.13 it is sufficient to show that there is a reflection between the category of prime event structures and the category of G-prime event structures.

First we define the category of prime event structures.

Definition 6.4.1
$\mathcal{PES}$ is the category which has prime event structures as its objects and *PES-morphisms* as its arrows.
A PES-morphism $f : (E_1, \leq_1, \#_1) \to (E_2, \leq_2, \#_2)$ is a partial function $f : E_1 \to E_2$ such that

(1) $\forall e \in E_1.\,(f(e) \text{ is defined } \Rightarrow \downarrow f(e) \subseteq f(\downarrow e))$

(2) $\forall e_1, e_2 \in E_1.\,((f(e_1) \text{ and } f(e_2) \text{ are defined and } f(e_1)\#_2 f(e_2)) \Rightarrow e_1 \#_1 e_2)$

(3) $\forall e_1, e_2 \in E_1.\,((f(e_1) \text{ and } f(e_2) \text{ are defined and } f(e_1) = f(e_2)) \Rightarrow (e_1 = e_2 \text{ or } \neg e_1 \#_1 e_2))$.

The identity morphism associated with an object is the identity function on its events; composition of PES-morphisms is composition of partial functions. □

In what follows it is sometimes convenient to use the following characterizations of the conflict relation of a prime event structure.

Lemma 6.4.2
Let P be a prime event structure. Then the following statements are equivalent:

(1) $\neg(e_1 \# e_2)$

(2) $\downarrow e_1 \cup \downarrow e_2 \in FC_P$

(3) $\exists c \in FC_P.\,\{e_1, e_2\} \subseteq c$. □

Proof.
(1) $\Rightarrow$ (2): If $\neg(e_1 \# e_2)$ then $\downarrow e_1 \cup \downarrow e_2$ is finite and downward-closed because $\downarrow e_1$ and $\downarrow e_2$ are finite (by (P2)) and downward-closed. Moreover, $\downarrow e_1 \cup \downarrow e_2$ is #-free by (P1), because both $\downarrow e_1$ and $\downarrow e_2$ are #-free by (P2).
The implications (2) $\Rightarrow$ (3) and (3) $\Rightarrow$ (1) are obvious. □

Similar to the situation for GPES-morphisms, also for PES-morphisms an alternative characterization in terms of the (finite) configurations can be given (see also [98]). This characterization is used as a definition for PES-morphisms in, e.g., [94, 98]. The proof is similar to the proof of Lemma 6.3.3.

Lemma 6.4.3

Let $P_1 = (E_1, \leq_1, \#_1)$ and $P_2 = (E_2, \leq_2, \#_2)$ be prime event structures and let $f : E_1 \to E_2$ be a partial function. Then f is a PES-morphism iff

(1') $\forall c \in C_{P_1} . f(c) \in C_{P_2}$

(2') $\forall c \in C_{P_1} . \forall e_1, e_2 \in c . ((f(e_1)$ and $f(e_2)$ are defined and $f(e_1) = f(e_2)) \Rightarrow e_1 = e_2)$. □

Proof.

Suppose that f is a PES-morphism. In order to prove (1'), let $c \in C_{P_1}$. By condition (2) in the definition of PES-morphism, $f(c)$ is $\#_2$-free because c is $\#_1$-free. Now let $e_2 \in E_2$ be such that $e_2 \leq_2 e_3$ for some $e_3 \in f(c)$. Then $e_3 = f(e_1)$ for some $e_1 \in c$. Hence condition (1) in the definition of PES-morphisms implies that $e_2 \in f(\downarrow e_1)$. Since c is downward-closed, we have $\downarrow e_1 \subseteq c$ and so $e_2 \in f(c)$. This proves that $f(c)$ is downward-closed, and hence $f(c) \in C_{P_2}$. Because each $c \in C_{P_1}$ is $\#_1$-free, condition (2') follows immediately from condition (3).

Now suppose that f satisfies the conditions (1') and (2') above. In order to prove condition (1) in the definition of PES-morphism, let $e_1 \in E_1$ be such that $f(e_1)$ is defined and suppose $e_2 \in \downarrow f(e_1)$, that is $e_2 \leq_2 f(e_1)$. Since $\downarrow e_1 \in C_{P_1}$, (1') implies $f(\downarrow e_1) \in C_{P_2}$. Hence $f(\downarrow e_1)$ is downward-closed and $e_2 \in f(\downarrow e_1)$. This proves condition (1).

In order to prove condition (2) in the definition of PES-morphism, suppose $f(e_1)$ and $f(e_2)$ are defined and $f(e_1) \#_2 f(e_2)$. Assume to the contrary that $\neg(e_1 \#_1 e_2)$. Then by Lemma 6.4.2 there exists $c \in C_1$ such that $\{e_1, e_2\} \subseteq c$. By condition (1') we also have that $f(c) \in C_2$. Because $\{f(e_1), f(e_2)\} \subseteq f(c)$, applying Lemma 6.4.2 leads to $\neg(f(e_1) \#_2 f(e_2))$, a contradiction. This proves that $e_1 \#_1 e_2$.

Finally, in order to prove condition (3) in the definition of PES-morphism, suppose $e_1, e_2 \in E_1$ are such that $f(e_1)$ and $f(e_2)$ are defined, $e_1 \neq e_2$, and $f(e_1) = f(e_2)$. Then by condition (2') there exists no $c \in C_{P_1}$ such that $\{e_1, e_2\} \subseteq c$ and hence Lemma 6.4.2 yields $e_1 \#_1 e_2$. □

Prime event structures can be viewed as G-prime event structures in which consistency of sets of events follows from the absence of pairwise conflicts.

For a prime event structure $P = (E, \leq, \#)$, define $pg(P) = (E, Con, \leq)$ where

$$Con = \{X \in P_F(E) \mid X \text{ is } \#\text{-free}\}.$$

Lemma 6.4.4

Let $P = (E, \leq, \#)$ be a prime event structure. Then $pg(P) = (E, Con, \leq)$ is a G-prime event structure with $C_P = C_{pg(P)}$.

Proof.

Condition (G0) follows from the irreflexivity of #. Condition (G1) follows immediately from (P2) and condition (G2) follows immediately from the definition of Con. In order to prove (G3), let $X \in Con$ and suppose $e' \leq e \in X$. Then $\neg(e'\#e'')$ for all $e'' \in X$ by (P1) because X is #-free. Moreover, $X \cup \{e'\}$ is finite because X is finite, and hence $X \cup \{e'\} \in Con$.

It is easy to see that $C_P = C_{pg(P)}$. □

In order to extend the map pg to a functor, we define again $pg(f) = f$ for each PES-morphism f.

Lemma 6.4.5

Let f be a PES-morphism from $P_1 = (E_1, \leq_1, \#_1)$ to $P_2 = (E_2, \leq_2, \#_2)$. Then $pg(f)$ is a GPES-morphism from $pg(P_1) = (E_1, Con_1, \leq_1)$ to $pg(P_2) = (E_2, Con_2, \leq_2)$.

Proof.

Condition (1) in the definition of GPES-morphism is satisfied, because it is the same as condition (1) in the definition of PES-morphism. In order to prove condition (2), let $X \in Con_1$. Then X is $\#_1$-free and hence $f(X)$ is $\#_2$-free by condition (2) in the definition of PES-morphism. Moreover, $f(X)$ is finite because X is finite. Hence we can conclude that $f(X) \in Con_2$. Finally, in order to prove condition (3), let $X \in Con_1$ and $e_1, e_2 \in X$ be such that $f(e_1)$ and $f(e_2)$ are defined and $e_1 \neq e_2$. Then $\neg(e_1\#_1 e_2)$ because X is $\#_1$-free. Condition (3) in the definition of PES-morphism now leads to $f(e_1) \neq f(e_2)$. □

The following result follows immediately from Lemma 6.4.4 and Lemma 6.4.5.

Theorem 6.4.6

pg is a functor from $\mathcal{PES}$ to $\mathcal{GPES}$. □

Conversely, a map from G-prime event structures to prime event structures is defined by interpreting absence of consistency of events as conflict.

For a G-prime event structure $G = (E, Con, \leq)$, define $gp(G) = (E, \leq, \#)$ where $\# \subseteq E \times E$ is given by

$$e_1 \# e_2 \Leftrightarrow \{e_1, e_2\} \notin Con.$$

Lemma 6.4.7

Let $G = (E, Con, \leq)$ be a G-prime event structure. Then $gp(G) = (E, \leq, \#)$ is a prime event structure with $FC_{gp(G)} = \{X \in P_F(E) \mid X$ is downward-closed and $\forall e_1, e_2 \in X. \{e_1, e_2\} \in Con\}$.

Proof.

First note that $\leq$ is a partial order and # is symmetric. By (G0), # is also irreflexive. In order to prove that $gp(G)$ satisfies (P1), let $e_0, e_1, e_2 \in E$ be such that $e_0 \# e_1 \leq e_2$. Then $\{e_0, e_1\} \notin Con$, and hence by (G2) $\{e_0, e_1, e_2\} \notin Con$. This implies by (G3) that $\{e_0, e_2\} \notin Con$. Hence $e_0 \# e_2$. This proves condition (P1). Condition (P2) in the definition of a prime event structure follows immediately from condition (G1).

From the definition of $gp(G)$ it is clear that $FC_{gp(G)} = \{X \in P_F(E) \mid X$ is downward-closed and $\forall e_1, e_2 \in X. \{e_1, e_2\} \in Con\}$. □

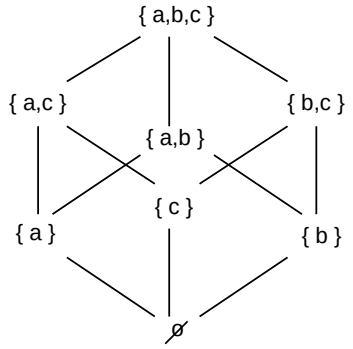


Figure 6.8: The prime event structure $gp(G)$

A G-prime event structure G may have a strictly smaller set of configurations than the prime event structure $gp(G)$. This is for instance the case for the G-prime event structure G depicted in Figure 6.6 for which $gp(G)$ is depicted in Figure 6.8.

Define $gp(f) = f$ for each GPES-morphism f.

Lemma 6.4.8
Let f be a GPES-morphism from G_1 to G_2. Then $gp(f)$ is a PES-morphism from $gp(G_1)$ to $gp(G_2)$.

Proof.
Immediate from the definitions of PES-morphism and GPES-morphism. □

Lemma 6.4.7 and Lemma 6.4.8 together imply the following.

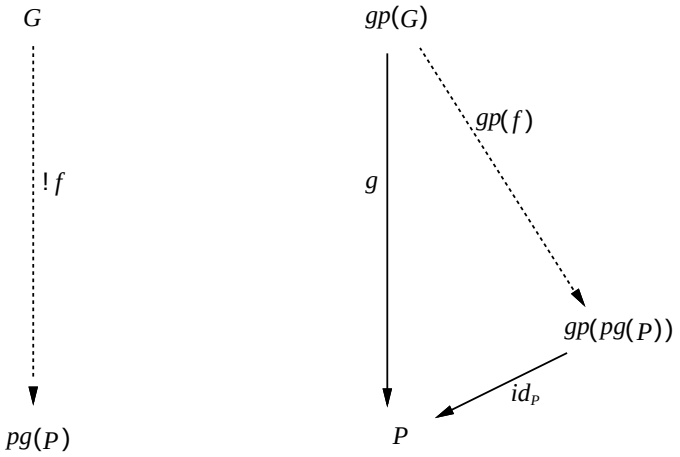
Theorem 6.4.9
gp is a functor from $\mathcal{GPES}$ to $\mathcal{PES}$. □

Next we prove that gp and pg form an adjunction.

Theorem 6.4.10
$gp : \mathcal{GPES} \rightarrow \mathcal{PES}$ and $pg : \mathcal{PES} \rightarrow \mathcal{GPES}$ form a reflection with gp the left adjoint and the identity arrows id_P as co-unit.

Proof.
First note that for each prime event structure P, $gp(pg(P)) = P$, and hence id_P is a PES-isomorphism from $gp(pg(P))$ to P.

Let $G = (E, Con, \leq)$ be a G-prime event structure, let $P = (E', \leq', \#')$ be a prime event structure, and let g be a PES-morphism from $gp(G) = (E, \leq, \#)$ to P. Then we must prove that there exists a unique GPES-morphism f from G to $pg(P) = (E', Con', \leq')$ such that the following diagram commutes.



By the definition of gp on arrows, it is sufficient to prove that g is a GPES-morphism from G to $pg(P)$. Because g satisfies condition (1) in the definition of PES-morphism, g also satisfies condition (1) in the definition of GPES-morphism. In order to prove that g satisfies condition (2) in the definition of GPES-morphism, suppose $X \in Con$. Then X is $\#$-free, and hence $g(X)$ is $\#'$-free by condition (2) in the definition of PES-morphism. This implies that $g(X) \in Con'$. Finally, in order to prove condition (3) in the definition of GPES-morphism, let $X \in Con$ and let $e_1, e_2 \in X$ be such that $g(e_1)$ and $g(e_2)$ are defined and $e_1 \neq e_2$. Then $\neg(e_1 \# e_2)$. Hence $g(e_1) \neq g(e_2)$ by condition (3) in the definition of PES-morphism. This proves that g is a GPES-morphism from G to $pg(P)$. □

In Section 5.6 a map pu from prime event structures to L-event structures has been defined. It is easy to see that this map is the same as the composition $gu \circ pg$. Thus by Lemma 6.4.4 and Lemma 6.3.6 the map pu associates an UL-event structure with each prime event structure.

Now define pu on morphisms by $pu(f) = gu \circ pg(f)$ for each PES-morphism f. Furthermore, denote $gp \circ ug$ by up. Then the following theorem follows immediately from Theorem 6.3.13 and Theorem 6.4.10.

Theorem 6.4.11

$up : \mathcal{ULES} \to \mathcal{PES}$ and $pu : \mathcal{PES} \to \mathcal{ULES}$ form a reflection with up the left adjoint and the identity arrows id_P as co-unit. □

Also in this case a full subcategory of $\mathcal{ULES}$ can be identified for which the reflection in Theorem 6.4.11 cuts down to an equivalence. We first identify the full subcategory of $\mathcal{GPES}$ for which the reflection in Theorem 6.4.10 cuts down to an equivalence.

To this aim we define for G-prime event structures $(E, Con, \leq)$ the following axiom expressing that consistency of sets can be recovered from pairwise consistency.

(PG) $\forall X \in P_F(E).\, ((\forall e, e' \in X.\, \{e, e'\} \in Con) \Rightarrow X \in Con)$. □

Lemma 6.4.12

$\mathcal{PES}$ is equivalent to the full subcategory of $\mathcal{GPES}$ the objects of which satisfy (PG).

Proof.

From the definition of pg it is clear that $pg(P)$ satisfies (PG) for every prime event structure P. By Theorem 6.4.10 it is then sufficient to prove that $pg(gp(G)) = G$ for every G-prime event structure G satisfying (PG).

Let $G = (E, Con, \leq)$ be a G-prime event structure which satisfies (PG) with $gp(G) = (E, \leq, \#)$ and $pg(gp(G)) = (E, Con', \leq)$. If $X \in Con$, then X is $\#$-free by (G2), and hence $X \in Con'$. Now suppose that $X \in Con'$. Then for all $e, e' \in X$, $\neg(e \# e')$, and hence $\{e, e'\} \in Con$. This implies that also $X \in Con$ by (PG). This proves that $Con = Con'$ and hence that $pg(gp(G)) = G$. □

Now we can characterize the full subcategory of $\mathcal{ULES}$ for which the reflection between $\mathcal{ULES}$ and $\mathcal{PES}$ from Theorem 6.4.11 cuts down to an equivalence. In this case the axiom (FC) used in Section 6.3 in the context of G-prime event structures is not strong enough. In the context of prime event structures it must be expressed that the union of *arbitrary* finite pairwise compatible sets of configurations is also a configuration. Thus we now define for UL-event structures $(E, C, \vdash)$ the following axiom (FC') which strengthens (FC).

(FC') $\forall D \in P_F(C).((\forall c, c' \in D. c \uparrow c') \Rightarrow \bigcup_{c \in D} c \in C)$.

Theorem 6.4.13

$\mathcal{PES}$ is equivalent to the full subcategory of $\mathcal{ULES}$ the objects of which satisfy (FC'), (BC), and (D).

Proof.

By Lemma 6.4.12 and Theorem 6.3.16 respectively, it is sufficient to prove that $ug(ES)$ satisfies (PG) for every UL-event structure ES satisfying (FC'), (BC), and (D), and that $gu(G)$ satisfies (FC') for every G-prime event structure G satisfying (PG).

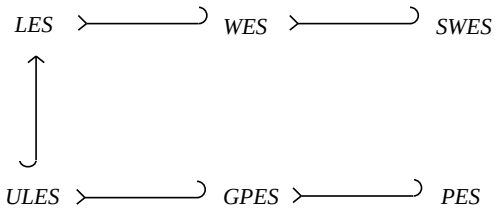
Let $ES = (E, C, \vdash)$ be an UL-event structure satisfying (FC'), (BC), and (D) with $ug(ES) = (E, Con, \leq)$ and let $X \in P_F(E)$ be such that for all $e, e' \in X$, $\{e, e'\} \in Con$. It must be proved that $X \in Con$. For all $e, e' \in X$ there exists $c \in C$ such that $\{e, e'\} \subseteq c$, and hence by the definition of $\leq$ also $\downarrow e \cup \downarrow e' \subseteq c$. By Lemma 6.3.14, $\downarrow e \in c$ for all $e \in X$. Now consider $D = \{\downarrow e \mid e \in X\}$. Then by (FC') $\bigcup_{c \in D} c \in C$. Because $X \subseteq \bigcup_{c \in D} c$, this implies that $X \in Con$. This proves that $ug(ES)$ satisfies (PG).

Now let $G = (E, Con, \leq)$ be a G-prime event structure satisfying (PG) with $gu(G) = (E, FC_G, \vdash)$ and suppose that $D \in P_F(FC_G)$ is such that for all $c, c' \in D$, $c \uparrow c'$. Then for all $e, e' \in \bigcup_{c \in D} c$, $\{e, e'\} \in Con$, and hence $\bigcup_{c \in D} c \in Con$ by (PG). Moreover, $\bigcup_{c \in D} c \in FC_G$ because each $c \in D$ is downward-closed. This proves that $gu(G)$ satisfies (FC'). □

6.5 Concluding Remarks

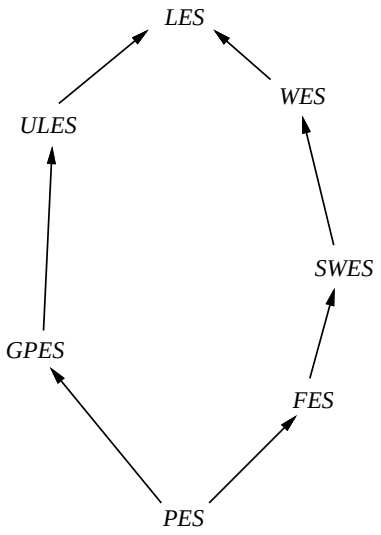
In this chapter we have investigated the relationship between the categories of L-event structures and UL-event structures introduced in the previous chapter and several

existing categories of event structures. In the following diagram our results are depicted. Recall that the co-reflection between $\mathcal{ULES}$ and $\mathcal{LES}$ has been proved in Corollary 5.7.11.



Thus L-event structures can be viewed as a generalization of (stable) W-event structures and UL-event structures can be viewed as a generalization of (G-)prime event structures. One way in which L-event structures generalize W-event structures is that the configuration structure may have more structure than given by set inclusion. Another way to generalize W-event structures is to put more structure in the notion of a configuration itself. This approach has been taken in [79] where configurations are posets of events.

Another interesting class of event structures not considered here is the class of *flow event structures* [14]. Flow event structures generalize prime event structures by relaxing the condition imposed upon the causality and conflict relation. In particular, condition (P1) in the definition of a prime event structure is dropped, the causal dependency relation need no longer be transitive, and the conflict relation does not have to be irreflexive. One of the motivations for working with flow event structures is that they seem to be more suitable for giving an event structure semantics for process calculi such as CCS. In [13] a map from the class $\mathcal{FES}$ of flow event structures to the class of stable W-event structures is defined. This map is injective and so $\mathcal{FES}$ can be viewed as a proper subclass of the class of stable W-event structures. The following diagram summarizes the relationship between the various classes of event structures (where we use the name of the category to denote the class of its objects). In this diagram an arrow between two classes denotes the fact that the given map from the first class to the second class is injective. Hence following the arrows we obtain more general classes of event structures. Note that from Corollary 6.2.8 and Theorem 6.3.16 it follows that in the framework of L-event structures the intersection of the classes $\mathcal{ULES}$ and $\mathcal{SWES}$ yields the class $\mathcal{GPES}$.



The classes of (G-)prime event structures and (stable) W-event structures also have an elegant characterization in terms of *domains* [85, 16]. The domain theoretic characterization of prime event structures has been given in [66]. Flow event structures yield the same class of domains [13]. Winskel has shown [96] that stable W-event structures yield the same class of domains as G-prime event structures. Finally, the domains corresponding to W-event structures have been characterized in [21], see also [96]. For L-event structures and UL-event structures however, it is not yet clear how one should go about obtaining a domain-theoretic characterization.

Chapter 7

Discussion

In this thesis we have investigated the behaviour of Petri nets at several levels of abstraction. In particular, we have investigated the transition system semantics from [63], a trace semantics, and an event structure semantics for Petri nets. These three types of semantics are increasingly more abstract. The semantic models used for representing the behaviour of Petri nets are generalizations of the models of transition systems, trace languages, and event structures, which have been used for giving a semantics for elementary net systems and 1-safe Petri nets. Also the semantic maps which have been defined are conservative extensions of the classical ones.

Each of these semantic maps abstracts from the distribution of a global state over local states. For the transition system semantics this is the only abstraction which is made. An important question is what are the corresponding models for the other two types of semantics that do not abstract from the distribution over local states. For the trace semantics this leads to the equivalence classes of occurrence sequences or processes from [9]. An interesting open problem is to find the counterpart in this sense of the event structure semantics for Petri nets in terms of local event structures introduced in this thesis. A solution to this problem would give some kind of unfolding of Petri nets generalizing the classical unfolding of 1-safe Petri nets. In contrast to the unfoldings considered in [61] and [24] such an unfolding would not be based on a colouring of tokens.

The following diagram gives an overview of the various models.

		linear time	*branching time*
distributed state	Petri nets	equivalence classes of processes	?
global state	multiset transition systems	local trace languages	local event structures

Whereas the universality of the transition system semantics and the trace semantics can be expressed through co-reflections with the category of Petri nets, a similar result does not hold for the event structure semantics in terms of local event structures. The problem is that due to auto-concurrency, the category of Petri nets is too rich in terms of objects *and* arrows. We have shown that by cutting down on the objects,

i.e. considering co-safe Petri nets, a co-reflection is obtained with the category of local event structures with the unique occurrence property. In order to get rid of the restriction of co-safeness, a proposal is given in order to lift local event structures to handle (finite) multisets of events. In this way an adjunction is obtained between the resulting category of event structures and the category of *all* Petri nets. The trouble with this more general approach is however that this adjunction is not a co-reflection. To solve this problem it seems that we must somehow find a way of distinguishing between multiple occurrences of the same transition due to auto-concurrency on the one hand and to causality on the other hand. It is however not at all obvious at present how this can be achieved.

The classes of trace languages and event structures which have been used for giving a semantics for Petri nets have been developed with the particular class of Petri nets in mind. However, we hope that the models will also turn out to be of independent interest. This hope is already partly justified by the results in Chapter 6 relating local event structures to other classes of event structures. In order to investigate the models, it would be interesting to look for universal constructions within the models.

The construction of the local event structure associated with a Petri net is essentially based on its runs represented as local traces. An interesting general construction has been used in [48] where a categorical construction is outlined, using an extension of the Yoneda embedding, in order to construct a behavioural description of a model out of its observations. It would be interesting to investigate what kind of objects would result from applying this construction to the observations of Petri nets represented as, e.g., local traces. This might then lead to an alternative branching time semantics for Petri nets.

Bibliography

[1] Aalbersberg, IJ. J., (1988), Studies in trace theory, PhD thesis, Leiden University.

[2] Aalbersberg, IJ. J., and Rozenberg, G., (1988), Theory of traces, Theoretical Computer Science 60, 1-82.

[3] Arnold, A., (1991), An extension of the notions of traces and of asynchronous automata, RAIRO, Theoretical Informatics and Applications 4, 355-393.

[4] Banâtre, J., and Le Métayer, D., (1993), Programming by multiset transformation, Communications of the ACM, Vol. 36, No. 1, 98-111.

[5] Barr, M., and Wells, C., (1990), Category Theory for Computing Science, Prentice Hall.

[6] Bednarczyk, M.A., (1988), Categories of asynchronous systems, PhD thesis, University of Sussex, report no. 1/88.

[7] Berry, G., and Boudol, G., (1992), The chemical abstract machine, Theoretical Computer Science 96, 217-248.

[8] Bertoni, A., Brambilla, M., Mauri, G., and Sabadini, N., (1981), An application of the theory of free partially commutative monoids: asymptotic densities of trace languages, Lecture Notes in Computer Science 118, 205-215.

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

[10] Best, E., Devillers, R., and Hall, J., (1992), The box calculus: a new causal algebra with multilabel communication, Lecture Notes in Computer Science 609, 21-69.

[11] Best, E., and Fernández C., C., (1988), Nonsequential Processes, EATCS Monographs on Theoretical Computer Science Vol. 13, Springer Verlag.

[12] Best, E., and Merceron, A., (1983), Discreteness, K-density and D-continuity of occurrence nets, Lecture Notes in Computer Science 145, 73-84.

[13] Boudol, G., (1990), Flow event structures and flow nets, Lecture Notes in Computer Science 469, 62-95.

[14] Boudol, G., and Castellani, I., (1988), Permutation of transitions: an event structure semantics for CCS and SCCS, Lecture Notes in Computer Science 354, 411-427.

[15] Cartier, P., and Foata, D., (1981), Problemes combinatoires de commutation et rearrangements, Lecture Notes in Mathematics 85.

[16] Davey, B.P., and Priestley, H.A., (1990), Introduction to lattices and order, Cambridge Mathematical Textbooks, Cambridge University Press.

[17] Degano, P., De Nicola, R., and Montanari, U., (1988), A distributed operational semantics for CCS based on Condition/Event systems, Acta Informatica 26, 59-91.

[18] Degano, P., Meseguer, J., and Montanari, U., (1989), Axiomatizing net computations and processes, Proc. of LICS 1989, 175-185.

[19] Devillers, R., (1993), Construction of S-invariants and S-components for refined Petri Boxes, Lecture Notes in Computer Science 691, 242-261.

[20] Diekert, V., and Rozenberg, G. eds., (1994), The Book on Traces, to appear.

[21] Droste, M., (1989), Event structures and domains, Theoretical Computer Science 68, 37-47.

[22] Ehrenfeucht, A., and Rozenberg, G., (1990), Partial (Set) 2-structures; Part II: State spaces of concurrent systems, Acta Informatica, v. 27, 343-368.

[23] Engberg, U.H., and Winskel, G., (1990), Petri nets as models of lineair logic, Lecture Notes in Computer Science 431, 147-161.

[24] Engelfriet, J., (1991), Branching processes of Petri nets, Acta Informatica, v. 28, 575-591.

[25] Engelfriet, J., (1993), A multiset semantics for the pi-calculus with replication, Lecture Notes in Computer Science 715, 7-21.

[26] Esparza, J., and Nielsen, M., (1994), Decidability issues for Petri nets - a survey, Bulletin of the European Association for Theoretical Computer Science 52, 244-262.

[27] Fernández, C. C., and Thiagarajan, P.S., (1984), D-continuous causal nets: a model of non-sequential processes, Theoretical Computer Science 28, 171-196.

[28] Foata, D., (1983), Rearrangements of words, in M. Lothaire, Combinatorics on words, Addison-Wesley.

[29] Gastin, P., (1990), Infinite traces, Lecture Notes in Computer Science 469, 277-308.

[30] Genrich, H.J., and Lautenbach, K., (1981), System modelling with high level Petri nets, Theoretical Computer Science 13, 109-136.

[31] Girard, J-Y, (1987), Linear logic, Theoretical Computer Science 50, 1-102.

[32] Goltz, U., (1988), On representing CCS programs by finite Petri nets, Lecture Notes in Computer Science 324, 339-350.

[33] Goltz, U., and Mycroft, A., (1984), On the relationship of CCS and Petri nets, Lecture Notes in Computer Science 172, 196-208.

[34] Goltz, U., and Reisig, W., (1983), The non-sequential behaviour of Petri nets, Information and Control 57, 125-147.

[35] Goltz, U., and Reisig, W., (1985), CSP-programs as nets with individual tokens, Lecture Notes in Computer Science 188, 169-196.

[36] Hack, M., (1976), Decidability questions for Petri nets, PhD thesis, M.I.T.

[37] Holt, A.W., and Commoner F., (1970), Events and conditions, Report of the Project MAC Conference on Concurrent Systems and Parallel Computation, 3-52.

[38] Holt, A.W., Saint, H., Shapiro, R., and Warshall, S., (1968), Final report of the Information Systems Theory Project, Technical Report RADC-TR-68-305, Rome Air Development Center, Griffis Air Force Base, New York.

[39] Hoogeboom, H.J., and Rozenberg, G., (1991), Diamond properties of elementary net systems, Fundamenta Informaticae Vol. XIV No. 3, 287-300.

[40] Hoogers, P.W., Kleijn, H.C.M., and Thiagarajan, P.S., (1992), A trace semantics for Petri nets, Lecture Notes in Computer Science 623, 595-604.

[41] Hoogers, P.W., Kleijn, H.C.M., and Thiagarajan, P.S., (1992), A trace semantics for Petri nets, Leiden University Techn. Rep. 92-03, to appear in Information and Computation.

[42] Hoogers, P.W., Kleijn, H.C.M., and Thiagarajan, P.S., (1993), Local event structures and Petri nets, Lecture Notes in Computer Science 715, 462-476.

[43] Hoogers, P.W., Kleijn, H.C.M., and Thiagarajan, P.S., (1993), An event structure semantics for general Petri nets, Leiden University Techn. Rep. 93-13, to appear in Theoretical Computer Science.

[44] Janicki, R., and Koutny, M., (1990), On some implementation of optimal simulation, DIMACS Series on Discrete Mathematics and Theoretical Computer Science 3, 231-250.

[45] Jantzen, M., (1986), Language theory of Petri nets, Lecture Notes in Computer science 254, 397-412.

[46] Jensen, K., (1981), Coloured Petri nets and the invariant method, Theoretical Computer Science 14, 317-336.

[47] Jensen, K., (1992), Coloured Petri nets: basic concepts, analysis methods and practical use; volume 1: basic concepts, EATCS Monographs on Theoretical Computer Science, Springer-Verlag.

[48] Joyal, A., Nielsen, M., and Winskel, G., Bisimulation and open maps, Proc. of LICS 1993, 418-427.

[49] Karp, R.M., and Miller, R.E., (1969), Parallel program schemata, Journal of Computer and System Sciences 3, 147-195.

[50] Kosaraju, S.R., (1982), Decidability of reachability in vector addition systems, Proc. of the 14th Annual ACM Symposium on Theory of Computing, San Francisco, California, 267-281.

[51] Lautenbach, K., (1972), Liveness in Petri nets, Internal Report GMD-ISF 72-02.1.

[52] Lodaya, K., Ramanujam, R., and Thiagarajan, P.S., (1989), A logic for distributed transition systems, Lecture Notes in Computer Science 354, 508-522.

[53] Maclane, S., (1971), Categories for the Working Mathematician, Graduate Texts in Mathematics, Springer-Verlag.

[54] Mayr, E.W., (1981), An algorithm for the general Petri net reachability problem, Proc. of the 13th Annual Symposium on Theory of Computing, 238-246.

[55] Mazurkiewicz, A., (1977), Concurrent program schemes and their interpretation, Aarhus University Techn. Rep. DAIMI PB-78.

[56] Mazurkiewicz, A., (1987), Trace theory, Lecture Notes in Computer Science 255, 279-324.

[57] Mazurkiewicz, A., (1988), Basic notions of trace theory, Lecture Notes in Computer Science 354, 285-363.

[58] Mazurkiewicz, A., (1989), Concurrency, modularity, and synchronization, Lecture Notes in Computer Science 379, 577-598.

[59] Meseguer, J., and Montanari, U., (1990), Petri nets are monoids, Information and Computation 88, 105-155.

[60] Meseguer, J., Montanari, U., and Sassone, V., (1992), On the semantics of place/transition Petri nets, Pisa University Techn. Rep. TR-27/92.

[61] Meseguer, J., Montanari, U., and Sassone, V., (1992), On the semantics of Petri nets, Lecture Notes in Computer Science 630, 286-301.

[62] Milner, R., Parrow, J., and Walker, D., (1992), A calculus of mobile processes, Information and Computation 100, 1-77.

[63] Mukund, M., (1992), Petri nets and step transition systems, International Journal of Foundations of Computer Science Vol. 3 No. 4, 443-478.

[64] Mukund, M., (1992), Transition system models for concurrency, Aarhus University Techn. Rep. DAIMI PB-399.

[65] Mukund, M., and Nielsen, M., (1993), CCS, locations and asynchronous transition systems, Lecture Notes in Computer Science 652, 328-341.

[66] Nielsen, M., Plotkin, G., and Winskel, G., (1981), Petri nets, event structures and domains, Part I, Theoretical Computer Science 13, 85-108.

[67] Nielsen, M., Rozenberg, G., and Thiagarajan, P.S., (1990), Behavioural notions for elementary net systems, Distributed Computing 4, 45-57.

[68] Nielsen, M., Rozenberg, G., and Thiagarajan, P.S., (1992), Elementary transition systems, Theoretical Computer Science 96, 3-33.

[69] Nielsen, M., Rozenberg, G., and Thiagarajan, P.S., (1992), Elementary transition systems and refinement, Acta Informatica 29, 555-578.

[70] Nielsen, M., Rozenberg, G., and Thiagarajan, P.S., (1993), Transition systems, event structures and unfoldings, to appear in Information and Computation.

[71] Olderog, E.-R., (1991), Nets, terms and formulas, Cambridge University Press, Cambridge.

[72] Peterson, J.L., (1976), Computation sequence sets, Journal of Computer and System Sciences 13, 1-24.

[73] Peterson, J.L., (1981), Petri net theory and the modelling of systems, Prentice-Hall, Englewood Cliffs, N.J.

[74] Petri, C.A., (1962), Kommunikation mit automaten, Schriften des Institutes für Instrumentelle Mathematik, Bonn.

[75] Petri, C.A., (1962), Fundamentals of a theory of asynchronous information flow, Information Processing 1962, Proceedings of the 1962 IFIP Congress, North-Holland Publishing Company Amsterdam.

[76] Petri, C.A., (1977), Non-sequential processes, GMD-ISF Report 77-05, Bonn.

[77] Pierce, B.C., (1991), Category Theory for Computer Scientists, The MIT Press.

[78] Reisig, W., (1985), Petri nets, an introduction, EATCS Monographs on Theoretical Computer Science Vol.4, Springer-Verlag.

[79] Rensink, A., (1992), Posets for configurations!, Lecture Notes in Computer Science 630, 269-285.

[80] Rozenberg, G., (1987), Behaviour of elementary net systems, Lecture Notes in Computer Science 254, 60-94.

[81] Rozenberg, G., and Thiagarajan, P.S., (1986), Petri nets: Basic notions, structure and behaviour, Lecture Notes in Computer Science 224, 585-668.

[82] Rozenberg, G., and Verraedt, R., (1983), Subset languages of Petri nets Part I: the relationship to string languages and normal forms, Theoretical Computer Science 26, 301-326.

[83] Rozoy, B., (1990), On distributed languages and models for distributed computation, Lecture Notes in Computer Science 469, 434-456.

[84] Rozoy, B., and Thiagarajan, P.S., (1991), Event structures and trace monoids, Theoretical Computer Science 91, 285-313.

[85] Scott, D.S., (1982), Domains for denotational semantics, Lecture Notes in Computer Science 140, 577-613.

[86] Shields, M.W., (1985), Concurrent machines, Computer Journal, v. 28, 449-465.

[87] Shields, M.W., (1992), Multitraces, hypertraces and partial order semantics, Formal Aspects of Computing Vol. 4 No. 6A, 649-672.

[88] Thiagarajan, P.S., (1987), Elementary net systems, Lecture Notes in Computer Science 254, 26-59.

[89] Thiagarajan, P.S., (1990), Some behavioural aspects of net theory, Theoretical Computer Science 71, 133-153.

[90] Vogler, W., (1990), Representation of a swapping class by one net, Lecture notes in Computer Science 424, 467-486.

[91] Vogler, W., (1991), A generalization of traces, RAIRO, Theoretical Informatics and Applications 2, 147-156.

[92] Winskel, G., (1980), Events in computation, Ph.D. Thesis, Dept. of Comp. Sc., University of Edinburgh.

[93] Winskel, G., (1982), Event structure semantics of CCS and related languages, Lecture Notes in Computer Science 140, 561-567.

[94] Winskel, G., (1984), Categories of models for concurrency, Lecture Notes in Computer Science 197, 246-267.

[95] Winskel, G., (1987), Petri nets, algebras, morphisms, and compositionality, Information and Computation 72, 197-238.

[96] Winskel, G., (1987), Event structures, Lecture Notes in Computer Science 255, 325-392.

[97] Winskel, G., (1988), An introduction to event structures, Lecture Notes in Computer Science 354, 364-397.

[98] Winskel, G., and Nielsen, M., (1992), Models for concurrency, to appear in S. Abramsky, D.M. Gabbay, T.S.E. Maibaum eds., Handbook of Logic in Computer Science.

Index

Overview Categories and Functors

Categories

Functors/Maps

Samenvatting

In dit proefschrift worden verschillende manieren bekeken om het gedrag van Petri netten te beschrijven. Het Petri net model is voortgekomen uit een theorie voor de beschrijving van informatiestromen met behulp van gedistribueerde toestanden en lokale toestandsovergangen. Deze theorie werd in 1962 door C.A. Petri geïnitieerd. Sindsdien hebben Petri netten sterk aan populariteit gewonnen, vooral doordat in dit model het gedistribueerde karakter van concurrente systemen op een natuurlijke manier kan worden gerepresenteerd. Bovendien blijken Petri netten een elegante onderliggende algebraïsche struktuur te hebben, waardoor het mogelijk is om een categorie van Petri netten te definiëren. Een voordeel van het beschouwen van modellen als categorieën is dat hiermee een formeel kader voorhanden komt om de struktuur van modellen te analyseren en om de onderlinge relaties tussen modellen te beschrijven. Zo kan men bijvoorbeeld het verschil in abstractie tussen twee modellen formaliseren als een co-reflectie tussen de twee bijbehorende categorieën, d.w.z. als een paar functors die tezamen een speciaal soort adjunctie vormen.

Voor eenvoudige net modellen zoals elementaire net systemen en 1-safe Petri netten is er een uitgebreide theorie voor het beschrijven van hun gedrag. Zo worden transitiesystemen gebruikt om te abstraheren van het gedistribueerde karakter van de toestanden in de netten. Verder worden Mazurkiewicz' traces gebruikt om de runs van zo'n net te beschrijven en prime event structures om de optredens van transities, de events, en hun onderlinge samenhang te beschrijven.

Bij algemene Petri netten echter kunnen zich situaties voordoen die niet goed uit te drukken zijn binnen deze modellen. Het is dan ook een niet-triviaal probleem om de semantiek van Petri netten te geven. Dit proefschrift beschrijft generalisaties van de bovenstaande gedragsmodellen waardoor het mogelijk wordt om de bestaande theorie uit te breiden tot het niveau van Petri netten. Een transitiesysteem semantiek voor Petri netten, nu geformuleerd in termen van multiset transitiesystemen is al voorgesteld door Mukund. Voor de andere twee benaderingen worden respectievelijk de gegeneraliseerde traces en de local event structures gebruikt die met dat doel in [41] en [43] zijn geïntroduceerd. In tegenstelling tot de meeste andere generalisaties van gedragsbeschrijvingen van eenvoudige net modellen tot Petri netten in de literatuur, is elk van de hier beschreven generalisaties strict. Zo wordt een beter inzicht verkregen in de gecompliceerde rol van concurrency in het gedrag van Petri netten.

Het proefschrift bestaat uit acht hoofdstukken waarin de drie verschillende manieren om het gedrag van Petri netten te beschrijven aan de orde komen en waarin achtergrond, samenhang en relatie met de literatuur worden besproken.

Allereerst wordt in de introduktie in Hoofdstuk 0 enige achtergrond gegeven met

betrekking tot de geschiedenis van Petri netten en het onderzoek van hun gedrag. In Hoofdstuk 1 worden enkele notaties ingevoerd, waarna in Hoofdstuk 2 Petri netten formeel worden geïntroduceerd.

In Hoofdstuk 3 wordt het model van multiset transitiesystemen besproken, niet alleen ter beschrijving van het gedrag van Petri netten, maar ook als kader waarbinnen een aantal fundamentele begrippen worden gedefinieerd. Door te abstraheren van de distributie van een toestand van een Petri net over lokale toestanden, kan het gedrag gerepresenteerd worden door zo'n multiset transitiesysteem. Formeel wordt de relatie tussen het model van Petri netten en het model van multiset transitiesystemen middels een resultaat van Mukund uitgedrukt als een co-reflectie tussen deze twee categorieën.

In Hoofdstuk 4 wordt een semantiek voor Petri netten bekeken gebaseerd op de runs van een Petri net. Hierbij wordt een beschrijving gegeven van de onderlinge samenhang van de optredens van de transities tijdens een executie. Omdat de conflicten tijdens een executie al zijn opgelost, wordt bij een dergelijke semantiek ook wel gesproken van een linear time semantiek. De traces van Mazurkiewicz worden gegeneraliseerd tot local traces, gedefinieerd met behulp van een onafhankelijkheidsrelatie die informatie bevat over het mogelijk concurrent optreden van multisets van acties in een bepaalde context. De relatie tussen het model van Petri netten en het resulterende trace model kan weer worden uitgedrukt als een co-reflectie tussen de categorieën.

Vervolgens wordt in Hoofdstuk 5 een branching time semantiek voor Petri netten bekeken waarbij bovendien een expliciet onderscheid wordt gemaakt tussen verschillende optredens van transities. Dit leidt tot een gedragsbeschrijving in termen van local event structures. De resulterende semantiek is echter enigszins beperkt, omdat geen rekening wordt gehouden met de mogelijkheid dat in sommige Petri netten transities concurrent met zichzelf kunnen optreden. Een gevolg hiervan is dat alleen een co-reflectie kan worden verkregen tussen een categorie van local event structures en de subcategorie van Petri netten, waarin auto-concurrency niet kan voorkomen.

In Hoofdstuk 6 wordt door middel van adjuncties een classificatie gegeven van enkele soorten event structures die in de literatuur zijn verschenen. Hieruit blijkt dat de in Hoofdstuk 5 geïntroduceerde local event structures ook formeel kunnen worden gezien als generalisaties van Winskels event structures en daardoor wellicht ook onafhankelijk van Petri netten nuttig kunnen zijn.

Tenslotte worden in de discussie in Hoofdstuk 7 de resultaten kort besproken.

Curriculum Vitae

De schrijver van dit proefschrift werd op 9 september 1966 te Amersfoort geboren. In 1984 behaalde hij het V.W.O.-diploma aan het Oranje Nassau College te Zoetermeer, waarna hij Informatica ging studeren aan de Rijksuniversiteit te Leiden. Deze studie werd in 1988 afgerond met een afstudeerproject op het gebied van Concurrency onder begeleiding van dr. H.C.M. Kleijn. Na het vervullen van de militaire dienstplicht trad de auteur op 1 december 1989 voor vier jaar in dienst van de Nederlandse Organisatie voor Wetenschappelijk Onderzoek (NWO). Gedurende deze vier jaar werd onder begeleiding van prof. dr. P.S. Thiagarajan, prof. dr. G. Rozenberg en dr. H.C.M. Kleijn aan de Rijksuniversiteit te Leiden het in dit proefschrift beschreven promotieonderzoek verricht als onderdeel van het project Research and Education in Concurrent Systems (REX). Sinds 11 juli 1994 is de auteur werkzaam bij het Nationaal Lucht- en Ruimtevaartlaboratorium te Vollenhove (Noordoostpolder).

Contents