# Formal Approach and Applications of Algebraic Higher-Order Nets

vorgelegt von
Diplom-Informatikerin
Kathrin Hoffmann

von der Fakultät IV - Elektrotechnik und Informatik
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktorin der Ingenieurwissenschaften
- Dr.-Ing. -

genehmigte Dissertation

Promotionsausschuss:

| | |
|---|---|
| Vorsitzender: | Prof. Dr. Günther Hommel |
| Berichter: | Prof. Dr. Hartmut Ehrig |
| Berichter: | Prof. Dr. Francesco Parisi-Presicce |

Tag der wissenschaftlichen Aussprache: 10. Oktober 2005

Berlin 2006
D 83

# Abstract

Approaches based on Petri nets are important for the development and redesign of distributed and concurrent systems. This thesis introduces the formal approach of algebraic higher-order nets as a novel modeling technique, which comprises the advantages of the well-researched net classes of coloured Petri nets and algebraic high-level nets. Moreover, algebraic higher-order nets can be seen as a formal approach for higher-order object nets, which are well-established for workflow modeling, but which so far have been mainly described informally. While the dynamic behavior of systems is graphically modeled by Petri nets, the concept of higher-order partial algebras turned out to be well-suited for algebraic higher-order nets. The combination of these techniques is achieved by the inscription of net elements with terms over the given data structure.

It is the aim of this thesis to develop a clear, mathematically well-founded, and practically valid formalism, which supports the flexibility and adaptability of models in an extensive way. In this context the extension by higher-order features is an important aspect, since we gain the possibility to abstract from functional behavior in the static structure of models. Hence, the functional behavior can be given at run time by suitable operations and altered dynamically by the exchange of these operations. The chosen concept of higher-order algebras highly increases the expressiveness of operations, so that even Petri nets become parameters and results of operations.

After a comprising introduction into higher-order algebras, the modeling technique of algebraic higher-order nets becomes formally defined and investigated extensively. As the main results of the theoretical part we have achieved horizontal structuring techniques and folding and unfolding constructions to support the composition of model segments on the one hand, and a compact description of abstract models on the other hand. In the practical part we present appropriate algebraic higher-order nets for the modeling of specific application domains, especially Petri nets and rules as tokens. Here we use rule-based transformations in the sense of graph transformation systems to attain a solid definition of rules and their applications.

The significance of our approach is demonstrated by several large examples and two case studies in the area of medical information systems and logistics processes.

# Zusammenfassung

Petri-Netz-basierte Ansätze spielen eine wichtige Rolle bei der Entwicklung und Neugestaltung von verteilten und nebenläufigen Systemen. Die vorliegende Arbeit stellt den formalen Ansatz der Algebraischen Higher-Order Netze als eine neue Modellierungstechnik vor, welche die Vorteile der gut untersuchten Netzklassen der Coloured Petri-Netze und Algebraischen High-Level Netze in sich vereint. Darüber hinaus sind Algebraische Higher-Order Netze als formaler Ansatz für Higher-Order Object Netze zu sehen, die sich in der Geschäftsprozessmodellierung bewährt haben, bisher aber überwiegend informell beschrieben wurden. Während das dynamische Verhalten der Systeme graphisch durch Petri-Netze modelliert wird, hat sich zur Beschreibung der Datenstrukturen das Konzept der Higher-Order Partiellen Algebren als besonders geeignet für Algebraische Higher-Order Netze erwiesen. Die Kombination dieser beiden Techniken wird durch die Beschriftung der Netzelemente mit Termen zur Datenstruktur erreicht.

Ziel ist es, einen klaren, mathematisch fundierten und praktisch einsetzbaren Formalismus zu entwickeln, der die Flexibilität und Anpassungsfähigkeit der Modelle in einem ausgedehnten Maße unterstützt. In diesem Zusammenhang ist die Erweiterung um Higher-Order Aspekte von wesentlicher Bedeutung. Dadurch ergibt sich die Möglichkeit, in der statischen Struktur der Modelle vom funktionalen Verhalten zu abstrahieren, welches erst zur Laufzeit durch die Angabe geeigneter Operationen konkretisiert wird und durch den Austausch dieser Operationen dynamisch verändert werden kann. Durch den gewählten Ansatz der Higher-Order Algebren erhöht sich die Ausdrucksmächtigkeit der Operationen enorm, so dass selbst Petri-Netze als Argumente und Resultate von Operationen verwendet werden können.

Nach einer umfassenden Einführung in das Konzept der Higher-Order Algebren, wird die Modellierungstechnik der Algebraischen Higher-Order Netze formal eingeführt und ausführlich untersucht. Als wesentliche Beiträge ergeben sich im theoretischen Teil horizontale Strukturierungstechniken und Konstruktionen zur (Ent-)Faltung von Netzen, um einerseits die Komposition von Modellsegmenten zu unterstützen und andererseits eine kompakte Beschreibung abstrakter Modelle zu erzielen. Im praktischen Teil werden geeignete Algebraische Higher-Order Netze für die Modellierung in speziellen Anwendungsgebieten vorgestellt. Hier sind im wesentlichen Petri-Netze und Regeln als Token zu nennen, wobei wir regelbasierte Transformation im Sinne von Graphtransformationssystemen verwenden, um eine mathematisch fundierte Beschreibung von Regeln und deren Anwendung zu erzielen.

Die Ausdrucksmächtigkeit von Algebraischen Higher-Order Netzen wird sowohl in einer Vielzahl von Beispielen als auch in zwei umfangreichen Fallstudien aus den Bereichen der medizinischen Informationssysteme und der Logistik demonstriert.

# Acknowledgements

The consistent help of my supervisor, Hartmut Ehrig, has been essentially encouraging for my early motivation and discussion with him has been extremely useful throughout the completion of this thesis.

Moreover I would like to express my sincere thanks to Julia Padberg for her professional guidance during my reseach work and to all members of the research group "Theoretical Computer Science/Formal Specification" for the positive work atmosphere.

Many thanks also to Till Mossakowski, Uwe Wolter, Francesco Parisi-Presicce, Lutz Schröder and Wolfgang Reisig for all the inspiring discussion and exchange.

I am especially grateful to Burkhard Beins for critically reading a draft of this thesis and for supporting and encouraging me all the way.

And a big thank you also to my mother for all her help and patience.

This work has been supported in part by the DFG-graduate college "Communication-based Systems" and by the European Research Training Networks "General Theory of Graph Transformation Systems (GETGRATS)" and "Syntactic and Semantic Integration of Visual Modelling Techniques (SegraVis)".

# Contents

# List of Figures

# Chapter 1

# Introduction

In the context of concurrent and distributed systems Petri nets, first introduced by C.A. Petri in [Pet62], are a well-known and widely used formalism and have been employed in practical applications in many different areas (see e.g. [Rei85, RT86, MOM89, MM90, Win87, Bau90]). Their graphical representation and formal semantics excellently support the modeling, simulation, and formal analysis of such systems. High-level net classes are obtained by combining Petri nets with an appropriate data type part. Despite the large variety of different high-level net classes, this thesis is based on three of them in particular.

Most prominent are coloured Petri nets [Jen92, Jen95, Jen97], a combination of Petri nets and a high-level programming language, which is an extension of the functional programming language Standard ML [MTHM97]. Coloured Petri nets offer formal verification methods and an excellent tool support, which has been used in numerous case studies within a large variety of different application areas.

Apart from this there are algebraic high-level nets [PER95, Pad96, EHP+02], which give rise to a formal and well-defined description due to their integration of classical algebraic specifications into Petri nets. Horizontal and vertical structuring techniques, formulated in the framework of category theory, are available for algebraic high-level nets, which provide the advantages of formal foundation, a proper degree of consistency, and compatibility results. Horizontal structuring techniques are given by the well-known union and fusion motivated by the constructions in [Jen81] for coloured Petri nets, while vertical structuring techniques are formalized in a rigorous way by rule-based transformations in the sense of high-level replacement systems [EHKP91]. Moreover, rules can be extended to property preserving rules, so that specific properties are preserved during the transformation [PU03].

With respect to the requirements of high-level net classes we are guided by the concept of higher-order object nets [LWH95, Han97], which are especially developed for workflow modeling and workflow system design. Higher-order object nets are based on higher-order nets [LH94], allowing functions, software components, and certain subnets as internal resources, and are enhanced by various constructions for practical issues. Thus, the research on higher-order object nets is focusing more on the methodology and infrastructure support.

In this thesis we investigate a new high-level net class called algebraic higher-order nets, which comprises all advantageous aspects of the above mentioned three of them. Although we do not want to claim that the net class of algebraic higher-order nets is the solution to all problems, we definitely achieve more flexibility concerning the support of some run-time aspects like operation late-binding mechanisms.

This thesis has been developed in close relation to the joint DFG research project "Petri Net Technology" between H. Weber (Coordinator), H. Ehrig (both from the Technical University Berlin), and W. Reisig (Humboldt-University of Berlin).

## 1.1 Requirements

In the following we discuss some requirements which should be provided by a net class to support structure flexibility and system adaptability in an extensive way.

**Composition of Models** The fact that Petri nets model concurrent and distributed systems is reflected in the distribution and integration of model segments to control the complexity of large models and the development of model segments by several teams at the same time. Thus, net classes require compositional techniques which support consistency integration and compatibility with other structuring techniques. Compositional techniques, allowing the formal construction of larger models from model segments with shared subparts called union, could be a good solution.

**Transformation of Models** During the process of development local refinement or local abstraction of subnets often becomes necessary. Moreover, the redesign or optimization of existing models refers to an addition or changing of model details. Very desirable are net classes, allowing on the one hand an easy and formal transformation of models, and ensuring on the other hand consistency and compatibility with other structuring techniques. Preservation of system properties during model transformation is an important aspect to avoid lengthy investigations. Rule-based transformation in the sense of high-level replacement systems and the concept of property preserving rules could help to solve the stated problem, since these approaches already have been worked out for different net classes and have been successfully applied to various case studies.

**Adaptive Models** The fixed part of a Petri net is given by the net structure, while the variable part is expressed by tokens. The fixed part can only be statically changed by using model transformations, while changes in the variable part result in an exchange of tokens. To support a rapidly changing environment and the reuse of existing models the variable part should be as large as possible. In this way a lot of work wrt. the model development can be saved and an immediate adaption of the model is realizable. The adaptability of models can be tackled in two different directions. On the one hand modeling techniques should separate the information of the baseline process from the information which depends on the current execution. In terms of Petri nets we seek a separation of the net structure from the data type part, so that the models can be on a high-level of abstraction. One aspect are operations which are carried out in the process and are normally fixed in the net structure. It is desirable to have an operation late-binding mechanism. Depending on a specific application the current execution of the process and the reaction to feedback should be given at run-time by adding and exchanging suitable operations as tokens. On the other hand there should be some constructions to develop abstract and flexible models, so that the level of abstraction will increase by using these constructions and a proper degree of prescription will be reached.

**Dynamic Reconfiguration** Model transformations on their own are often not enough to handle dynamic model reconfigurations, because the process has to be constantly modified according to previous events and a fixed specification of a "one-size-for-all" process is almost impossible. Furthermore, a running process may be influenced by some abnormal situations causing deviations from the pre-defined model. But the rearrangement of processes should neither interfere the overall net structure nor interrupt the running system. Dynamic reconfiguration is referred to as ad-hoc modification of models at run time. To cope effectively with the

dynamics of modifying models the applications of the so-called token game and transformations should be independent from each other. Moreover, these activities should be interleaved to allow changes of the net structure while the system is still running and a highly reusability of existing models should be supported.

**Compact Description of Models** In general, classical Petri nets lead to very large models, because the variable part is restricted to black tokens. A considerable step forward are high-level nets due to the integration of a data type part into Petri nets. To obtain a more compact description folding constructions, which transfer specific parts of the process specification into the data type part and identify distinguished elements, are in demand.

**Dynamic Tokens** From the Petri net perspective real-world objects appear in the form of tokens. During the last decade tokens have been considered as more and more complex data elements. In contrast to ordinary black tokens and (passive) data elements, dynamic tokens are nowadays considered to have their own individual behavior like Petri nets, as tokens themselves. Dynamic tokens behave like ordinary tokens, i.e. they may move through the system. But in contrast to them they may also change their state, while being moved through the system. Thus, the data type part of high-level net classes has to be powerful enough to consider dynamical tokens in the system.

**Rule Tokens** As mentioned above rule-based transformation of structures is an important aspect. Depending on the application domain these structures appear as tokens in the system. Thus, rules and transformations should be covered by the data type part of high-level net classes to get rules as tokens and transformation as appropriate operations. This would lead to abstract models, which are flexible in the sense that different transformations are not attached to the net structure in a fixed way, but could be expressed by a set of rule tokens. In this way changes of specific transformations result in an exchange of rule tokens.

**Integration of Software Components** Since software components are subject to frequent updating and expansion, local modifications or substitutions of software components should be easily integrated into the existing model. Moreover, some software components like commercial software tools may be license-restricted and can only be shared in a limited way. To solve these problems software components should be encapsulated into data elements, which represent executable functions or programs and can be used as tokens in the corresponding model. Flexibility in the sense of local modification or substitution of software components could be expressed in this way by exchanging the relevant tokens. In the case of license-restricted software components the number of tokens could represent the number of licenses and the access authorization of different tasks to software components could be reflected in the net structure.

## 1.2 Motivation and Main Results

In this thesis we investigate the new high-level net class of algebraic higher-order nets, which satisfies most, but not yet all requirements presented in the previous section. Those which are not satisfied yet will be discussed as future work in Chapter 10.

We suggest that the data type part in high-level nets should be extended to include function spaces, product types, and partiality, so that some inherent problems of high-level nets like operation late-binding mechanisms can be solved. As a

result, functions and more complex objects, which are composed by functions like Petri nets themselves, are introduced into the models as tokens. Moreover, they can be manipulated by higher-order functions because they are not bound to transitions once and for all. Thus we obtain a more compact description of models but also a flexible and adaptable system structure to cope with evolutionary and exceptional changes.

The new contribution of this thesis is the concept of algebraic higher-order nets. Algebraic higher-order nets are an extension of algebraic high-level nets [PER95, Pad96, EHP+02], because the data type part is defined by the concepts of higher-order algebras instead of classical algebras. Thus we follow this approach by giving a set theoretic definition of domains and operations. Classical algebras are a powerful tool to specify the most common data structures used in programming languages and their semantics. But the expressive power of the equational logic used to axiomatize the data structures is not always strong enough to specify the intended data structure by a finite set of equations, for instance for Petri nets and rules as tokens. In order to obtain also an algebraic specification of these data structures we need algebraic higher-order specifications and have to move from equational to conditional equational logic. However, the formalism of algebraic high-level nets is adequate in application domains, where the context is known from the very beginning, so that the system can be modeled by a Petri net with a fixed net structure. In other application domains like business process modeling it is also desirable to support a rapidly changing environment, but, using the formalism of algebraic high-level nets, changes of the environment can only be modeled by changing the fixed net structure. By contrast, algebraic higher-order nets gain more flexible models due to the higher-order features introduced by the data type part. Therefore, models do not necessarily have to be completely specified during the design process, since the intended behavior can be still given at run time. Additionally, we are able to reach an even higher level of abstraction through the application of constructions, which transfer specific parts of the fixed net structure into the variable part of the model.

Algebraic higher-order nets can be seen as a formal approach of higher-order nets [LH94, Han97], where the data type part is defined within an algebraic framework to introduce higher-order concepts into Petri nets. Higher-order nets provide a formalism as a basis for the workflow language of higher-order object nets [LWH95, Han97], which excellently supports the design of adaptive and configurable workflow systems and incorporates mechanisms for run time communication between workflow models and their surrounding environment. But up to now there is only a basic formalism available for higher-order nets and it is desirable to have a fully worked out theory in order to fill the gap between the theoretical approach of higher-order nets and the practical approach of higher-order object nets.

In some respect algebraic higher-order nets are orthogonal to coloured Petri nets [Jen92, Jen95, Jen97], because we prefer an intensional approach of higher-order algebras. In an intensional setting we have the advantage of function equivalence testing within some models and we can distinguish between different functions which exhibit the same behavior. By contrast, extensional equality of functions means that two functions are equal if they always produce the same results for the same arguments. Standard ML [MTHM97], the data type part of coloured Petri nets, cannot implement equality on function types. This means that it would be difficult to consider functions as first-class citizens and thus tokens in coloured Petri nets. Apart from this, the direct use of a programming language for the data type part can be seen as a methodological drawback, because it is essential for the development process, that we capture the distinction between an abstract description of what the system must do and how it will be realized and implemented. In our approach of algebraic higher-order nets the data type part given within an algebraic framework

is more abstract than the corresponding programs. This effort of abstraction is not only beneficial for the reuse, evolution, and maintenance of data structures, but also for validation and verification aspects. Moreover, formal specification environments offer the possibility of a code generator.

In this thesis we formally investigate the notions and results of higher-order algebras. Although some of them had already been published in [Wol05], we have revised and extended them to achieve several results for our purpose. Based on this data type we define the basic formalisms of algebraic higher-order net schemes and algebraic higher-order nets, where an algebraic higher-order net is an algebraic higher-order net scheme together with a suitable model. We generalize these approaches at a categorical level which especially involves the treatment of mappings between algebraic higher-order net schemes resp. algebraic higher-order nets. Those mapping are called morphisms, which are essential for the formulation of structuring techniques. We define the formal semantics for algebraic higher-order nets not only by the firing step semantics, but also by a generalization of the well-known notion of process semantics from low-level to higher-order nets.

The main results of this thesis on the field of algebraic higher-order nets include the following. Some preliminary work has been already published in [Hof00, HM02, Hof03, HMPP04, HEM05].

- Structuring techniques, called union and fusion, supporting composition of algebraic higher-order net schemes and identification of subnets within an algebraic higher-order net scheme.
- Preservation of the firing behavior by algebraic higher-order net morphisms.
- Folding and unfolding constructions wrt. constant symbols, supporting the flexibility and adaptability of models and preserving the firing behavior.
- Folding and unfolding constructions wrt. product types, leading to more compact descriptions of models and preserving the firing behavior.

Besides these theoretical aspects, this thesis gives a detailed motivation for algebraic higher-order nets from a practical point of view. For this reason we feature the following valuable classes of algebraic higher-order nets, each of them for specific application areas.

- Algebraic higher-order nets with nets and interaction relations as tokens for applications in the area of workflow systems, agent-oriented approaches, and flexible manufacturing systems.
- Algebraic higher-order nets with nets and rules as tokens for applications in all areas where dynamic changes of the token net structure have to be considered while the system is still running, notably flexible workflows and medical information systems.
- Algebraic higher-order nets with graph rules and rules for the manipulation of graph rules as tokens for applications in the area of mobile policies.

In addition to several large examples this thesis includes two case studies in the area of medical information systems and logistics processes to demonstrate the structural flexibility and system adaptability within the practical use of algebraic higher-order nets. To cope with aspects of higher-order specifications we discuss some extensions of algebraic higher-order nets, which are an important step towards the implementation of our concepts.

## 1.3 Structure of the Thesis

The thesis is structured as follows. Since algebraic higher-order nets are an extension of algebraic high-level nets, we review the main concepts of classical algebras and algebraic high-level nets in Chapter 2. Section 2.1 includes the notions of place/transitions nets and place/transition systems as well as the results of rule-based transformations for place/transition systems, which is used not only in the example in Section 2.4 to illustrate algebraic high-level nets with place/transition systems and rules as tokens, but also in the subsequent sections concerning different classes of algebraic higher-order nets for specific application domains.

Chapter 3 supplies the description of the basic formalism of algebraic higher-order net schemes and algebraic higher-order nets. First we formally investigate the notions and results of higher-order algebras in Section 3.1. Moreover, we point out special features introduced by higher-order algebras and some design decisions which are useful for our approach of algebraic higher-order nets. Then we give the necessary notions of algebraic higher-order net schemes and algebraic higher-order nets and define their operational behavior in Section 3.2. To illustrate the use of algebraic higher-order nets Section 3.3 contains a simple example. Without going into detail the overall motivation for certain folding constructions and structuring techniques is mentioned as well.

Chapter 4 presents algebraic higher-order net schemes and algebraic higher-order nets in a categorical setting. An essential result of this chapter is that horizontal structuring techniques for algebraic higher-order net schemes are given by the notions of union and fusion, motivated by the constructions of [Jen81], but here achieved in a categorical way by the (finite) cocompleteness of the category of algebraic higher-order net schemes. The second main result shows that algebraic higher-order nets related by mappings exhibit the same firing behavior. In Section 4.3 we propose a notion of higher-order processes. This idea is based on the notions in [EHP+02] for algebraic high-level nets, where the well-known concept of low-level process became generalized to high-level processes. The theory presented in this chapter is a good starting point for further research in the field of algebraic higher-order nets, especially to achieve further structuring techniques and to obtain compatibility results (see Chapter 10).

The main goal of Chapter 5 is the introduction of folding and unfolding constructions, which are new in the area of high-level nets and arise due to the higher-order features of algebraic higher-order net schemes. The first main result of this chapter shows that folding and unfolding constructions wrt. constant symbols preserve the operational behavior but the folding construction realizes a more abstract level and supports mechanisms of operation late-binding. Note that in the context of higher-order algebras operations can be represented as constants of an appropriate function type. The concept of unfolding wrt. constant symbols has a counterpart in the context of high-level nets, where the flattening construction is used to define the semantics of a high-level net by a classical Petri net (see e.g. [Jen92, Hum89, Gen86, Pad96]). Vice versa the concept of folding is related to the step from low to high-level nets. Because product types are available in higher-order algebras, we can give a folding construction wrt. product types. The second main result of this chapter shows, that both nets are equivalent wrt. their operational behavior, but the folding construction allows a more compact representation. Analogously, we give an unfolding construction wrt. product types, which preserves the firing behavior. Last but not least we prove in this chapter that both folding and unfolding constructions are inherently inverse.

In Chapter 6 we distinguish between three different classes of algebraic higher-order nets, each of them for specific application areas. The main idea is to present a specific higher-order signature together with a corresponding higher-order algebra

in order to achieve sorts and operations suitable for particular application areas. To describe the main idea and to demonstrate the practical use of these valuable net classes we present several large examples. In detail we distinguish between the following net classes. First we capture the idea of elementary object systems [Val98, Val01] and present algebraic higher-order nets with nets and interaction relations as tokens. Through elementary object systems the paradigm "nets as tokens" became introduced in order to allow nets as tokens, called object nets, within a net, called a system net. The object nets can move through a system net and, this is the interesting aspect, they can interact with both, the system net and with other object nets. As we show exemplarily, elementary object systems can be translated into semantically equivalent algebraic higher-order net and interaction systems, while the formal definition of this translation is an aspect of future work. Next we introduce algebraic higher-order nets with nets and rules as tokens. This concept is based on the idea of algebraic high-level nets with nets and rules as tokens presented in [HEM05] (see also Section 2.4). Here we exploit the approach of rule-based transformations to change the token net structure by the application of rule tokens. This concept is especially used in the case study of a medical information system in Chapter 8. Finally we present algebraic higher-order nets for applications in the area of mobile policies, which already have been published in [HMPP04]. Here, the tokens are on the one hand graph rules to represent policies and on the other hand rules for the manipulation of mobile policies because policies have to be changed when they migrate from site to site to adapt to external requirements of specific domains.

To obtain algebraic higher-order specifications we introduce the concept of conditional equational logic in Section 7.1. As shown in [Wol05] this formalism provides initial/free semantics, which is essential for the existence of term generated models. Moreover, we discuss the approach of HasCasl [SM02] in Section 7.2. HasCasl has been introduced as a higher-order extension of the first-order algebraic specification language Casl (Common Algebraic Specification Language) [Mos04]. It is geared to the specification of functional programs, in particular in Haskell. Because tools for HasCasl have already been implemented, this is a first step towards an implementation of our approach of algebraic higher-order nets. To illustrate the concept of HasCasl we present several HasCasl-specifications including a detailed specification of place/transitions-systems and a parametrized specification of rule-based transformations, which can be instantiated by several categories.

An outline of the case study in the area of medical information systems demonstrates the use of algebraic higher-order nets with nets and rules as tokens in Chapter 8. Applications of both, horizontal structuring techniques and folding constructions, are presented within the case study in the area of logistics processes in Chapter 9.

Our future work is manyfold and is summarized in Chapter 10. We discuss vertical structuring techniques and compatibility of these structuring techniques with the horizontal structuring techniques in Section 10.1. Moreover, the relation of algebraic higher-order nets to other net classes, especially algebraic high-level nets, is still an open question (see Section 10.2). In Section 10.3 we outline the main concept of dynamic reconfigurations based on the combination of folding and unfolding construction wrt. constant symbols and rule-based transformations. In order to evaluate and improve the applicability of our concept of algebraic higher-order nets we discuss several aspects supported by tools in Section 10.4.

Finally, Chapter 11 contains a summary of the achieved results and a conclusion. This chapter is followed by appendices containing notions frequently used in this thesis and the basic notions of category theory, free commutative monoids, and elementary object nets.

In the following we would like to give some advice when reading this thesis. If the reader is already familiar with place/transition systems and algebraic high-level

nets, Chapter 2 may be skipped. We assume that after reading the description of the basic formalism and the introductory example of algebraic higher-order nets in Chapter 3, the reader is well-prepared to follow the different net classes of algebraic higher-order nets for specific application domains in Chapter 6 and the case studies in Chapter 8 and Chapter 9. For those readers particularly interested in the theoretical aspects, respectively structuring techniques and (un-)folding construction, we can recommend Chapter 4, Chapter 5, and Chapter 10.

## 1.4 Related Work

The work related to this thesis can be divided into two different areas. On the one hand there are several approaches to capture higher-order features in an algebraic approach. A detailed discussion and comparison of classical and higher-order algebraic approaches will be given in Section 3.1. On the other hand, besides the high-level net classes of coloured Petri nets [Jen92, Jen95, Jen97], algebraic high-level nets [PER95, Pad96, EHP$^+$02], and higher-order (object) nets [LH94, LWH95, Han97] mentioned above, there are several net classes closely related to algebraic higher-order nets.

The integration of algebraic specifications methods into Petri nets has been investigated for instance in [Vau86, Hum89, Rei91, Lil95]. The combination of predicate-transition nets [GL81] and many-sorted partial algebras is presented in [Krä89, Sch89], where the extension of the first order framework by function and product types is briefly discussed. But the authors feel that further research and a deeper analysis of these concepts are necessary.

The idea of executable routines (called jobs) attached to transitions has been first introduced by Function nets [God83]. The jobs are executed whenever their transitions fire. Function nets and their variants, such as Funsoft nets [Gru91, DG91, DG94], have been used for instance in information system modeling and for the simulation of database machine architectures. By these approaches functions themselves, however, could not be manipulated as resources, because they are bound to their transitions once and for all.

Elementary object systems [Val98, Val01] provides a two-level modeling technique, in which Petri nets themselves are considered as token objects and communicate by synchronizing transitions. In [Kum01, Kum02] this approach is extended to reference nets, which allow the dynamic creation of an arbitrary number of net instances during the execution of models. Reference nets are labeled by Java expressions and provide the concept of synchronous channels to support an n-level modeling technique. Independently, [BBPP04] presents the concept of Petri hypernets, where net tokens migrate within the tree-like hierarchy, i.e. the level on which a net token appears can be changed. While in all these approaches the token net structure cannot be changed during transition firing, controlled transformations of token nets is discussed in the context of linear logic Petri nets [Far99, Far00] and feature structure nets [Wie01]. The transformations are carried out by transition guards, which are fixed in the net structure. Hence, in these approaches a transition can only perform a specific transformation.

Although in this thesis we do not need features of object-oriented modeling like inheritance, encapsulation, and dynamic binding, this would be an interesting aspect to extend our approach by integration of these features. [ADCR01] is a good overview of different kinds of high-level net classes, which integrate object-oriented modeling and Petri nets.

# Chapter 2

# Review of Algebraic High-Level Nets

In this chapter we review some main concepts frequently used in this thesis. This allows a detailed comparison of the existing notions with our new concept of algebraic higher-order nets. First we review the main concepts of place/transitions nets and place/transition systems in Section 2.1. Then we introduce the basic theory for rule-based transformations of place/transition systems. This theory is inspired by graph transformation systems [Ehr79, Roz97], which have already been generalized to net transformations systems in [EHKP91, EP04], including high-level and low-level nets. The theory in these papers is based on pushouts in the corresponding categories according to the double-pushout approach of graph transformations in [Ehr79]. In Section 2.2 we recall the necessary notions of classical algebraic specifications [EM85]. They are well established as formal specifications of abstract data types and software systems. Moreover, they are a suitable data type part for high-level nets. The combination of Petri nets and classical algebraic specifications leads to the concept of algebraic high-level nets [PER95, Pad96], which gives rise to a formal and well-defined description of concurrent and distributed systems. The notions of algebraic high-level nets are reviewed in Section 2.3, where we use the terminology of [EHP+02]. In Section 2.4 we show how algebraic high-level nets can be used to model the requirements of the "House of Philosophers", which is a small system inspired by the case study "the Hurried Philosophers" [SB01]. In order to model nets and rules as tokens we present a specific signature together with a corresponding algebra with specific sorts for place/transition systems and rules. Moreover, there are operations corresponding to the firing of transitions and applying a rule to a place/transition system, respectively. Since algebraic high-level nets are based on classical algebraic specifications we are able to give a set theoretic definition of domains and operations. Most of the notions and results have been already published in [HEM05].

## 2.1 Place/Transition Systems

In this section we review the notion of place/transition nets and place/transition systems, which is a place/transition net with an initial marking. As net formalism we use the idea of an algebraic formulation of Petri nets and following the notation of "Petri nets are Monoids" in [MM90]. Moreover, we employ the category of place/transitions systems to introduce the basic theory for rule-based transformations as presented in [HEM05]. In order to improve the intuition of our concepts for the reader we give in this section an explicit approach of rule-based transforma-

tions for place/transition systems, which extends the theory of place/transition net transformations taking into account also initial markings, and avoids categorical terminology like pushouts.

**Definition 2.1.1 (Place/Transition Nets)**
A place/transition (P/T) net $N = (P, T, pre, post)$ consist of

- a set of places $P$ and a set of transitions $T$ and
- pre- and post conditions $pre, post : T \longrightarrow P^{\oplus}$ assigning to each transition $t \in T$ an element of the free commutative monoid $P^{\oplus}$ over the set $P$ of places with binary operation $\oplus$.

**Remark 2.1.2 (Free Commutative Monoid)**
An element $M$ of the free commutative monoid $P^{\oplus}$ can be represented as the linear sum $M = \sum_{i=1}^{n} k_i \cdot p_i$ with coefficients $k_i \in \mathbb{N}$. We say that $\sum_{i=1}^{n} k_i \cdot p_i$ is in normal form if all $k_i \neq 0$ and $p_i$ are pairwise distinct. The notion of commutative monoids corresponds to multisets, i.e. $M \in P^{\oplus}$ can also be considered as function $M : P \longrightarrow \mathbb{N}$ with finite support and represented by a formal sum $\sum_{p \in P} M(p) \cdot p$. For $M_1, M_2 \in P^{\oplus}$ $M_1 \leq M_2$ if and only if for all coefficients $M_1(p) \leq M_2(p)$. Addition/subtraction of elements is defined as componentwise addition/subtraction of coefficients, where subtraction $M_1 \ominus M_2$ is only defined if $M_1 \geq M_2$. Finally, $p \in M_1$ if and only if $M_1(p) > 0$. The formal definitions of free commutative monoids and their operations are listed in Appendix B.

**Definition 2.1.3 (Firing Behavior of P/T-nets)**
Given a P/T-net $N = (P, T, pre, post)$, then

1. a marking of $N$ is given by $M \in P^{\oplus}$,
2. a transition $t \in T$ is $M$-enabled for a marking $M \in P^{\oplus}$, denoted by $M[t\rangle$, if we have $pre(t) \leq M$ and
3. if $t \in T$ is $M$-enabled the follower marking $M'$ is given by

$$M' = M \ominus pre(t) \oplus post(t)$$

and denoted by $M[t\rangle M'$.

**Definition 2.1.4 (Place/Transition Net Morphisms)**
Given P/T-nets $N_1 = (P_1, T_1, pre_1, post_1)$ and $N_2 = (P_2, T_2, pre_2, post_2)$, a P/T-net morphism $f : N_1 \longrightarrow N_2$ is given by $f = (f_P, f_T)$ with functions

$$f_P : P_1 \longrightarrow P_2 \text{ and } f_T : T_1 \longrightarrow T_2$$

satisfying that $f$ is compatible with pre- and post domain, i.e. the following diagram commutes componentwise:

$$\begin{array}{ccc} T_1 & \overset{pre_1}{\underset{post_1}{\rightrightarrows}} & P_1^{\oplus} \\ {\scriptstyle f_T}\downarrow & = & \downarrow{\scriptstyle f_P^{\oplus}} \\ T_2 & \overset{pre_2}{\underset{post_2}{\rightrightarrows}} & P_2^{\oplus} \end{array}$$

where $f_P^{\oplus} : P_1^{\oplus} \longrightarrow P_2^{\oplus}$ is the unique homomorphic extension of $f_P : P_1 \longrightarrow P_2$ (see Appendix B).

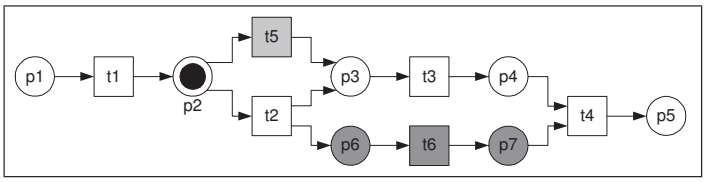



Figure 2.1: P/T-system

**Definition 2.1.5 (Category PTNet)**
The category defined by P/T-nets as objects and P/T-net morphisms as morphisms is denoted by **PTNet** where the composition of P/T-net morphisms is defined componentwise for places and transitions.

For details about this category see e.g. [MM90].

**Definition 2.1.6 (Place/Transition Systems)**
A place/transition (P/T) system $PN = (P, T, pre, post, M^0)$ consists of

- a P/T-net $(P, T, pre, post)$ and
- an initial marking $M^0 \in P^{\oplus}$.

An example of a P/T-system can be found in Fig. 2.1.

**Definition 2.1.7 (Place/Transition System Morphisms)**
Given P/T-systems $PN_i = (P_i, T_i, pre_i, post_i, M_i^0)$ for $i \in \{1, 2\}$, a P/T-system morphism $f : N_1 \longrightarrow N_2$ is given by a P/T-net morphism $f = (f_P, f_T)$ with functions $f_P : P_1 \longrightarrow P_2$ and $f_T : T_1 \longrightarrow T_2$ additionally satisfying that the initial marking of $N_1$ at place $p$ is smaller or equal to that of $N_2$ at $f_P(p)$:

$$f_P^{\oplus}(M^0_{1|p}) \leq M^0_{2|f_P(p)} \text{ for all } p \in P_1.$$

The restriction $M^0_{1|p}$ is defined by $M^0_{1|p} = M_1^0(p) \cdot p$ where $M_1^0$ is considered as function. Moreover, a P/T-system morphism $f$ is called strict if $f_P^{\oplus}(M^0_{1|p}) = M^0_{2|f_P(p)}$ and $f_P, f_T$ are injective.

**Definition 2.1.8 (Category PTSys)**
The category defined by P/T-systems as objects and P/T-system morphisms as morphisms is denoted by **PTSys** where the composition of P/T-system morphisms is defined componentwise for places and transitions.

The next step in order to define transformations of P/T-systems is to define the gluing of P/T-systems in analogy to concatenation in the string case.

**Definition 2.1.9 (Gluing of P/T-Systems)**
Given P/T-systems $PN_i = (P_i, T_i, pre_i, post_i, M_i^0)$ for $i \in \{0, 1, 2\}$ with strict inclusion $inc : PN_0 \longrightarrow PN_1$ and P/T-system morphism $f : PN_0 \longrightarrow PN_2$. Then the gluing $PN_3$ of $PN_1$ and $PN_2$ via $(PN_0, f)$, written $PN_3 = PN_1 +_{(PN_0,f)} PN_2$, is defined by the following diagram (1), called "gluing diagram", with

1. $\forall p \in P_1 = P_0 \uplus (P_1 \setminus P_0) : f'_P(p) = \underline{\text{if}}\ p \in P_0\ \underline{\text{then}}\ f_P(p)\ \underline{\text{else}}\ p$
   $\forall t \in T_1 = T_0 \uplus (T_1 \setminus T_0) : f'_T(t) = \underline{\text{if}}\ t \in T_0\ \underline{\text{then}}\ f_T(t)\ \underline{\text{else}}\ t$

2. $PN_3 = (P_3, T_3, pre_3, post_3, M_3^0)$ with
   - $P_3 = P_2 \uplus (P_1 \setminus P_0)$, $T_3 = T_2 \uplus (T_1 \setminus T_0)$,
   - $pre_3(t) =$ if $t \in T_2$ then $pre_2(t)$ else $f_P'^{\oplus}(pre_1(t))$,
   - $post_3(t) =$ if $t \in T_2$ then $post_2(t)$ else $f_P'^{\oplus}(post_1(t))$ and
   - $M_3^0 = M_2^0 \oplus (M_1^0 \ominus M_0^0)$.

$$\begin{array}{ccc} PN_0 & \xrightarrow{inc} & PN_1 \\ {\scriptstyle f}\downarrow & (1) & \downarrow{\scriptstyle f'} \\ PN_2 & \xrightarrow[inc']{} & PN_3 \end{array}$$

**Remark 2.1.10 (Disjoint Union)**
The disjoint union in the definition of $P_3$ and $T_3$ takes care of the problem that there may be places or transitions in $PN_2$, which are - by chance - identical to elements in $P_1 \setminus P_0$ or $T_1 \setminus T_0$, but only elements in $PN_0$ and $f(PN_0)$ should be identified. In this case the elements of $P_1 \setminus P_0$ and $T_1 \setminus T_0$ should be renamed before applying the construction above.

**Fact 2.1.11 (Gluing of P/T-Systems)**
The gluing $PN_3 = PN_1 +_{(PN_0,f)} PN_2$ is a well-defined P/T-system such that $f' : PN_1 \longrightarrow PN_3$ is a P/T-system morphism, $inc' : PN_2 \longrightarrow PN_3$ is a strict inclusion and the gluing diagram (1) commutes, i.e. $f' \circ inc = inc' \circ f$.

**Proof:**

1. $PN_3$ is a well-defined P/T-system, because $pre_3, post_3 : T_3 \longrightarrow P_3^{\oplus}$ are well-defined functions. Now $f' = (f_P', f_T') : PN_1 \longrightarrow PN_3$ is a P/T-system morphism because we have $pre_3 \circ f_T' = f_P'^{\oplus} \circ pre_1$ (and similar for $post$) by case distinction:

   **Case 1** For $t \in T_0$ we have $pre_3(f_T'(t)) = pre_3(f_T(t)) = pre_2(f_T(t)) = f_P^{\oplus}(pre_0(t)) = f_P'^{\oplus}(pre_0(t)) = f_P'^{\oplus}(pre_1(t))$.

   **Case 2** For $t \in T_1 \setminus T_0$ we have $pre_3(f_T'(t)) = pre_3(t) = f_P'^{\oplus}(pre_1(t))$.

   We have marking compatibility of $f'$ by:

   **Case 1** For $p \in P_0$ we have
   $f_P'^{\oplus}(M_{1|p}^0) = f_P^{\oplus}(M_{0|p}^0) \leq M_{2|f_P(p)}^0 \leq M_{3|f_P(p)}^0 = M_{3|f_P'(p)}^0$.

   **Case 2** For $p \in P_1 \setminus P_0$ we have
   $f_P'^{\oplus}(M_{1|p}^0) = f_P'^{\oplus}((M_1^0 \ominus M_0^0)_{|p}) = (M_1^0 \ominus M_0^0)_{|p} \leq M_{3|f_P'(p)}^0$.

2. $inc' : PN_2 \longrightarrow PN_3$ is a P/T-system inclusion by construction. The marking $M_3^0$ is well-defined because $M_0^0 \leq M_1^0$ and $M_{0|p}^0 = M_{1|p}^0$ for $p \in P_0$ by strict inclusion $inc : PN_0 \longrightarrow PN_1$. Moreover $inc'$ is strict, because we have $M_1^0 \ominus M_0^0 \in (P_1 \setminus P_0)^{\oplus}$ which implies for $p \in P_2$ $M_{2|p}^0 = M_{3|p}^0$.

3. $f' \circ inc = inc' \circ f$ by construction.

□

**Remark 2.1.12 (Gluing Diagram)**
The gluing diagram (1) is a pushout diagram in the category **PTSys**. This implies that the transformation of P/T-systems defined below is in the spirit of the double-pushout approach for graph transformations and high-level replacement systems (see [Ehr79, EHKP91]).

Two examples of gluing and gluing diagrams are given in Fig. 2.8 (see Section 2.4), where $G = L_2 +_{I_2} C$ and $H = R_2 +_{I_2} C$ in the left hand and the right hand gluing diagram respectively. Our next goal is to define rules, application of rules and transformations of P/T-systems.

**Definition 2.1.13 (Rule of P/T-Systems)**
A rule $r = (L \xleftarrow{i_1} I \xrightarrow{i_2} R)$ of P/T-systems consists of P/T-systems $L$, $I$, and $R$, called left-hand side, interface, and right-hand side of $r$ respectively, and two strict P/T-system morphisms $I \xrightarrow{i_1} L$ and $I \xrightarrow{i_2} R$ which are inclusions.

**Remark 2.1.14 (Application of Rules)**
The application of a rule $r$ to a P/T-system $G$ is given by a P/T-system morphism $L \xrightarrow{m} G$, called match. Now a direct transformation $G \overset{r}{\Longrightarrow} H$ via $r$ can be constructed in two steps. In a first step we construct the context $C$ given by $(G - m(L)) \cup m \circ i_1(I)$ and P/T-system morphisms $I \xrightarrow{c} C$ and $C \xrightarrow{c_1} G$, where $c_1$ is a strict inclusion. This means we remove the match $m(L)$ from $G$ and preserve the interface $m \circ i_1(I)$. In order to ensure that $C$ becomes a subsystem of $G$ we have to require a "gluing condition" (see Def. 2.1.15). This makes sure that $C$ is a P/T-system and we have $m \circ i_1 = c_1 \circ c$ in the "context diagram" (1). In a second step we construct $H$ as gluing of $C$ and $R$ along $I$, this means we obtain the gluing diagram (2) from $I \xrightarrow{c} C$ and $I \xrightarrow{i_2} R$.

$$\begin{array}{ccccc} L & \xleftarrow{i_1} & I & \xrightarrow{i_2} & R \\ {\scriptstyle m}\downarrow & (1) & \downarrow{\scriptstyle c} & (2) & \downarrow{\scriptstyle n} \\ G & \xleftarrow[c_1]{} & C & \xrightarrow[c_2]{} & H \end{array}$$

Now we define the gluing condition and the context construction.

**Definition 2.1.15 (Gluing Condition)**
Given a strict inclusion morphism $i_1 : I \longrightarrow L$ and a P/T-system morphism $m : L \longrightarrow G$ the gluing points $GP$, dangling points $DP$ and the identification points $IP$ of $L$ are defined by

$$\begin{array}{ll} GP = & P_I \cup T_I \\ DP = & \{p \in P_L | \exists t \in (T_G \setminus m_T(T_L)) : m_P(p) \in pre_G(t) \oplus post_G(t)\} \\ IP = & \{p \in P_L | \exists p' \in P_L : p \neq p' \wedge m_P(p) = m_P(p')\} \\ & \cup \{t \in T_L | \exists t' \in T_L : t \neq t' \wedge m_T(t) = m_T(t')\} \end{array}$$

where $p \in P_L = \sum_{i=1}^{n} k_i \cdot p_i$ means $p = p_i$ and $k_i \neq 0$ for some $i$. Then the gluing condition is satisfied if all dangling and identifications points are gluing points, i.e. $DP \cup IP \subseteq GP$.

**Fact 2.1.16 (Context P/T-System)**
Given a strict inclusion $i_1 : I \longrightarrow L$ and a P/T-system morphism $m : L \longrightarrow G$, then the following context P/T-system $C$ is well-defined and leads to the following commutative diagram (1), called "context diagram", if the gluing condition $DP \cup IP \subseteq GP$ is satisfied. $C = (P_C, T_C, pre_C, post_C, M_C^0)$ is defined by

$$\begin{array}{l} P_C = (P_G \setminus m_P(P_L)) \cup m_P(P_I), \\ T_C = (T_G \setminus m_T(T_L)) \cup m_T(T_I), \\ pre_C = pre_{G|C}, post_C = pre_{G|C}, \text{ and} \\ M_C^0 = M_{G|C}^0. \end{array} \qquad \begin{array}{ccc} I & \xrightarrow{i_1} & L \\ {\scriptstyle c}\downarrow & (1) & \downarrow{\scriptstyle m} \\ C & \xrightarrow[c_1]{} & G \end{array}$$

The morphisms in (1) are defined by $c : I \longrightarrow C$ to be the restriction of $m : L \longrightarrow G$ to $I$ and $c_1 : C \longrightarrow G$ to be a strict inclusion.

**Proof:** The P/T-system $C$ and $pre_C, post_C : T_C \longrightarrow P_C^{\oplus}$ with $pre_C = pre_{G|C}$ and $post_C = pre_{G|C}$ are well-defined if $DP \cup IP \subseteq GP$. For $t \in T_C$ we have to show $pre_C(t) \in P_C^{\oplus}$ (and similar for $post_C(t)$).

**Case 1** For $t \in T_G \setminus m_T(T_L)$ we have $pre_C(t) = pre_G(t) = \sum_{i=1}^{n} k_i \cdot p_i$. Assume $p_i \notin P_C$ for some $i \leq n$. Then $p_i \in m_P(P_L) \setminus m_P(P_I)$ with $p_i \in pre_G(t)$. Hence there is $p_i' \in P_L \setminus P_I$ with $m_P(p_i') = p_i$. This implies $p_i' \in DP$ and $p_i' \notin GP$ and contradicts the gluing condition $DP \cup IP \subseteq GP$.

**Case 2** For $t \in m_T(T_I)$ we have $t' \in T_I$ with $t = m_T(t')$. This implies

$$\begin{aligned} pre_C(t) = \;& pre_G(t) = pre_G(m_T(t')) = m_P^{\oplus}(pre_L(t')) \\ = \;& m_P^{\oplus}(pre_I(t')) \in m_P^{\oplus}(P_I^{\oplus}) = (m_P(P_I))^{\oplus} \subseteq P_C^{\oplus}. \end{aligned}$$

Moreover $c : I \longrightarrow C$ satisfies the marking condition in Def. 2.1.7, because this is true for $m : L \longrightarrow G$ and $c$ is restriction of $m$. Finally $c_1 : C \longrightarrow G$ is a strict inclusion by construction. This leads to the commutative diagram (1) in **PTSys**. □

**Remark 2.1.17 (Pushout Diagram)**
Note that we have not used the "identification condition" $ID \subseteq GP$, which is part of the gluing condition. But this is needed to show that the context diagram (1) is - up to isomorphism - also a gluing diagram and hence a pushout diagram in the category **PTSys**. This means that $C$ is constructed in such a way that $G$ becomes the gluing of $L$ and $C$ via $I$, i.e. $G \cong L +_I C$.

An example of a context diagram is the left diagram in Fig. 2.8 (see Section 2.4), where $C$ is the context P/T-system for $i_2 : I_2 \longrightarrow L_2$ and $g : L_2 \longrightarrow G$.

Now a direct transformation is given by the combination of a context diagram and a gluing diagram.

**Definition 2.1.18 (Applicability of Rules and Transformation)**
A rule $r = (L \xleftarrow{i_1} I \xrightarrow{i_2} R)$ is called applicable at match $L' \xrightarrow{m} G$ if $L = L'$ and the gluing condition is satisfied for $i_1$ and $m$. In this case we obtain a context P/T-system $C$ with context diagram (1) and a gluing diagram (2) with $H = C +_I R$ leading to a direct transformation $G \overset{r}{\Longrightarrow} H$ consisting of the following diagrams (1) and (2). A (rule-based) transformation $G \overset{*}{\Longrightarrow} H$ is a sequence of direct transformations $G = G_0 \overset{r_1}{\Longrightarrow} G_1 \overset{r_2}{\Longrightarrow} \ldots \overset{r_n}{\Longrightarrow} G_n = H$ with $G = H$ for $n = 0$.

$$\begin{array}{ccccc} L & \xleftarrow{i_1} & I & \xrightarrow{i_2} & R \\ {\scriptstyle m}\downarrow & (1) & \downarrow{\scriptstyle c} & (2) & \downarrow{\scriptstyle n} \\ G & \xleftarrow[c_1]{} & C & \xrightarrow[c_2]{} & H \end{array}$$

An example for a direct transformation is given in Fig. 2.8 (see Section 2.4).

**Remark 2.1.19 (Double-Pushout Approach)**
As pointed out in Remark 2.1.12 and Remark 2.1.17 already the context diagram (1) and the gluing diagram (2) are pushout diagrams in the category **PTSys**. Hence a direct transformation $G \overset{r}{\Longrightarrow} H$ is given by the two pushouts (1) and (2), also called double pushout (DPO). In the DPO-approach of graph transformations (see [Ehr79]), high-level replacement systems [EHKP91] and Petri net transformations [EP04] a direct transformation is defined by a DPO-diagram. For P/T-systems our definition is equivalent up to isomorphism to the existence of a DPO in the category **PTSys**.

## 2.2 Classical Algebras

In this section we review definitions and results of algebraic specifications based on the notation of [EM85]. Conceptually, such specifications consist of sorts, operation symbols, and axioms, where the axioms are equations. The semantics of algebraic specifications are defined by a class of classical algebras with total functions. We present a slightly different version to [EM85] in this section. These differences concerns the variables, which are normally locally bound to the axioms. Here we choose a global approach of variables, which is more suitable for the data type part of algebraic high-level nets.

**Definition 2.2.1 (Signature)**
A classical signature $\Sigma = (S, OP, scr, tar)$ is given by a set $S$ of sort symbols and a set $OP$ of operation symbols, a source function $scr : OP \longrightarrow S^*$, and a target function $tar : OP \longrightarrow S$. Here, $S^*$ is the set of finite (including empty) sequences of elements of $S$.

We write $op : s_1 \dots s_n \to s$ for $op \in OP$ with $src(op) = s_1 \dots s_n \in S^*, n \geq 0$, and $tar(op) = s \in S$. The notion $c :\to s$ is used to indicate that $src(c) = \lambda$ (the empty string) and denotes a constant symbol.

**Definition 2.2.2 (Signature with Variables)**
Let $\Sigma = (S, OP, scr, tar)$ be a signature, $X$ a set, called set of variables, with $X \cap OP = \emptyset$ and $var : X \longrightarrow S$ a function, called sorts of variables. Then $\Sigma = (S, OP, scr, tar, X, var)$ is a signature with variables. We write $x : s$ for $x \in X$ and $var(x) = s$. To simplify the notation we also denote a signature with variables by the following three components $(S, OP, X)$.

**Definition 2.2.3 (Signature Morphism)**
Given signatures $\Sigma_1 = (S_1, OP_1, X_1)$ and $\Sigma_2 = (S_2, OP_2, X_2)$, a signature morphism $f_\Sigma : \Sigma_1 \longrightarrow \Sigma_2$ is a tuple of functions

$$f_\Sigma = (f_S : S_1 \longrightarrow S_2, f_{OP} : OP_1 \longrightarrow OP_2, f_X : X_1 \longrightarrow X_2)$$

such that the following diagram commutes componentwise:

$$\begin{array}{ccccccc} S_1^* & \xleftarrow{src_1} & OP_1 & \xrightarrow{tar_1} & S_1 & \xleftarrow{var_1} & X_1 \\ \downarrow{\scriptstyle f_S^*} & = & \downarrow{\scriptstyle f_{OP}} & = & \downarrow{\scriptstyle f_S} & = & \downarrow{\scriptstyle f_X} \\ S_2^* & \xleftarrow[src_2]{} & OP_2 & \xrightarrow[tar_2]{} & S_2 & \xleftarrow[var_2]{} & X_1 \end{array}$$

where $f_S^* : S_1^* \longrightarrow S_2^*$ denotes the extension of $f_S : S_1 \longrightarrow S_2^\star$ to sequences recursively defined by

$$f_S^*(\lambda) = \lambda \text{ and}$$
$$f_S^*(s_1 \dots s_n) = f_S(s_1) \dots f_S(s_n) \text{ for } s_1 \dots s_n \in S_1^*, n \geq 1.$$

**Definition 2.2.4 (Set of Terms)**
Let $\Sigma = (S, OP, X)$ be a signature. The set of terms of sort $s \in S$ is the least set $T_{\Sigma,s}(X)$ inductively constructed by the following rules:

**1.** $$x : s \in X \Rightarrow x \in T_{\Sigma,s}(X)$$

**2.** $$(c :\to s) \in OP \Rightarrow c \in T_{\Sigma,s}(X)$$

**3.** $$(op : s_1 \dots s_n \to s) \in OP, term_1 \in T_{\Sigma,s_1}(X), \dots, term_n \in T_{\Sigma,s_n}(X)$$
$$\Rightarrow op(term_1, \dots, term_n) \in T_{\Sigma,s}(X).$$

The set of $\Sigma$-terms $T_\Sigma(X)$ is defined by $T_\Sigma(X) = \bigcup_{s \in S} T_{\Sigma,s}(X)$.

**Definition 2.2.5 (Translation of Terms)**
Let $f_\Sigma : \Sigma_1 \longrightarrow \Sigma_2$ be a signature morphism with $f_\Sigma = (f_S, f_{OP}, f_X)$. The translation of $\Sigma_1$-terms by $f_\Sigma$ to $\Sigma_2$-terms is obtained by replacing the corresponding variables and operation symbols, i.e. $f^\sharp_\Sigma : T_{\Sigma_1}(X_1) \longrightarrow T_{\Sigma_2}(X_2)$ is recursively defined by

**1.** $$f^\sharp_\Sigma(x) = f_X(x) \text{ for all } x : s \in X_1$$

**2.** $$f^\sharp_\Sigma(c) = f_{OP}(c) \text{ for all } (c :\rightarrow s) \in OP_1$$

**3.** $$f^\sharp_\Sigma(op(term_1, \ldots, term_n)) = f_{OP}(op)(f^\sharp_\Sigma(term_1), \ldots, f^\sharp_\Sigma(term_n))$$
$$\text{for all } (op : s_1 \ldots s_n \rightarrow s) \in OP_1,$$
$$term_1 \in T_{\Sigma_1,s_1}(X_1), \ldots, term_n \in T_{\Sigma_1,s_n}(X_1).$$

**Remark 2.2.6 (Translation as Free Construction)**
The translation of terms can be achieved as a free construction over $X_1$ in **Sets**, if $f_X : X_1 \longrightarrow X_2$ is injective, i.e. $f_X^{-1} \circ f_X = id_{X_1}$.

**Definition 2.2.7 (Algebras)**
A (classical) $\Sigma$-algebra $A = (A(S), A(OP))$ of a signature $\Sigma = (S, OP, X)$ is given by a $S$-sorted set $A(S) = (A_s)_{s\in S}$, the carrier of $A$, and by an $OP$-sorted set $A(OP) = (op_A)_{op\in OP}$ of operations, i.e. for every $op : s_1 \ldots s_n \rightarrow s \in OP$ there is a (total) function $op_A : A_{s_1} \times \ldots \times A_{s_n} \longrightarrow A_s$.

**Definition 2.2.8 (Term Evaluation)**
Let $\Sigma = (S, OP, X)$ be a signature. A variable valuation of $X$ in a $\Sigma$-algebra $A$ is a (total) function $v : X \longrightarrow A$ with $v(x) \in A_s$ for $x : s \in X$. The term evaluation of $T_\Sigma(X)$ in $A$ wrt. $v$ is a (total) function $v^\sharp : T_\Sigma(X) \longrightarrow A$ recursively defined by

**1.** $$v^\sharp(x) = v(x) \text{ for all } x : s \in X$$

**2.** $$v^\sharp(c) = c_A \text{ for all } (c :\rightarrow s) \in OP$$

**3.** $$v^\sharp(op(term_1, \ldots, term_n)) = op_A(v^\sharp(term_1), \ldots, v^\sharp(term_n))$$
$$\text{for all } (op : s_1 \ldots s_n \rightarrow s) \in OP, term_1 \in T_{\Sigma,s_1}(X), \ldots, term_n \in T_{\Sigma,s_n}(X).$$

**Definition 2.2.9 (Homomorphisms)**
Let $\Sigma = (S, OP, X)$ be a signature. For two $\Sigma$-algebras $A_1$ and $A_2$ a homomorphism is given by a (total) $S$-sorted function

$$f_A : A_1 \longrightarrow A_2 = (f_{A,s} : A_{1,s} \longrightarrow A_{2,s})_{s\in S}$$

satisfying the following two conditions:

1. for each constant symbol $c :\rightarrow s \in OP$ we have $f_{A,s}(c_A) = c_B$ and
2. for each operation symbol $op : s_1 \ldots s_n \rightarrow s \in OP$ and all $a_i \in A_{s_i}$ for $i \in \{1, \ldots, n\}$ we have $f_{A,s}(op_A(a_1, \ldots, a_n)) = op_B(f_{A,s_1}(a_1), \ldots, f_{A,s_n}(a_n))$.

A homomorphism $f_A : A_1 \longrightarrow A_2$ is called isomorphism, if for $s \in S$ all functions $f_{A,s} : A_{1,s} \longrightarrow A_{s,2}$ are bijective.

**Definition 2.2.10 (Category Alg($\Sigma$))**
The category **Alg($\Sigma$)** consists of $\Sigma$-algebras as objects and homomorphisms as morphisms. The identity is the identity homomorphism and composition is associative.

**Definition 2.2.11 (Equations and Validity)**
Given a signature $\Sigma = (S, OP, X)$. Let $s \in S$ and $term_l, term_r \in T_{\Sigma,s}(X)$. Then

1. $(term_l = term_r)$ is called an equation of sort $s$ wrt. $\Sigma$.
2. An equation $(term_l = term_r)$ is called valid in a $\Sigma$-algebra $A$ if for all variable valuations $v : X \longrightarrow A$ we have

$$v^\sharp(term_l) = v^\sharp(term_r).$$

Let $f_\Sigma = (f_S, f_{OP}, f_X) : \Sigma \longrightarrow \Sigma'$ be a signature morphism. Then any equation $(term_l, term_r)$ of sort $s$ wrt. $\Sigma$ can be uniquely translated into an equation $(f^\sharp_\Sigma(term_l) = f^\sharp_\Sigma(term_r))$ of sort $f_S(s)$ wrt. $\Sigma'$.

**Definition 2.2.12 (Algebraic Specification)**
An algebraic specification $SP = (\Sigma, E)$ consists of a signature $\Sigma = (S, OP, X)$ and a set $E$ of equations over the signature $\Sigma$. Let $A$ be a $\Sigma$-algebra. Then $A$ is a $SP$-algebra if all equations $e \in E$ are valid in $A$.

**Definition 2.2.13 (Category Alg($SP$))**
The category **Alg($SP$)** consists of $SP$-algebras as objects and homomorphisms as morphisms.

**Definition 2.2.14 (Algebraic Specification Morphism)**
Given two algebraic specifications $SP_i = (\Sigma_i, E_i)$ for $i \in \{1, 2\}$. A signature morphism $f_\Sigma : \Sigma_1 \longrightarrow \Sigma_2$ is called specification morphism, written $f_{SP} : SP_1 \longrightarrow SP_2$, if for each equation $e \in E_1$ the translated equation $f^\sharp_\Sigma(e)$ is provable from $E_2$ with the equation calculus in [EM85].

**Definition 2.2.15 (Category SP)**
The category **SP** consists of algebraic specifications as objects and algebraic specification morphisms as morphisms.

**Definition 2.2.16 (Forgetful Functor $V_{f_{SP}}$)**
Given a specification morphism $f_{SP} : SP_1 \longrightarrow SP_2$ with corresponding signature morphism $f_\Sigma = (f_S, f_{OP}, f_X) : \Sigma_1 \longrightarrow \Sigma_2$, the forgetful functor

$$V_{f_{SP}} : \mathbf{Alg(SP}_2\mathbf{)} \longrightarrow \mathbf{Alg(SP}_1\mathbf{)}$$

is defined for all $SP_2$-algebras $A_2$ by $V_{f_{SP}}(A_2) = A_1 \in \mathbf{Alg(SP}_1\mathbf{)}$ with

$$\begin{array}{ll} A_{1,s} = A_{2,f(s)} & \text{for all } s \in S_1 \\ op_{A_1} = f_{OP}(op)_{A_2} & \text{for all } op \in OP_1 \end{array}$$

and for all $SP_2$-homomorphism $f_{2,A} : A_2 \longrightarrow B_2$ by $V_{f_{SP}}(f_{2,A}) = f_{1,A} : A_1 \longrightarrow B_1$ with

$$f_{1,A,s} = f_{2,A,f_S(s)} \text{ for all } s \in S_1.$$

**Definition 2.2.17 (Free Functor $F_{f_{SP}}$)**
Given a specification morphism $f_{SP} : SP_1 \longrightarrow SP_2$, then the forgetful functor $V_{f_{SP}} : \mathbf{Alg(SP}_2\mathbf{)} \longrightarrow \mathbf{Alg(SP}_1\mathbf{)}$ has a left adjoint functor

$$F_{f_{SP}} : \mathbf{Alg(SP}_1\mathbf{)} \longrightarrow \mathbf{Alg(SP}_2\mathbf{)},$$

i.e. there is an adjunction $F_{f_{SP}} \vdash V_{f_{SP}}$.

For further details of Def. 2.2.16 and Def. 2.2.17 we refer to [EM85].

## 2.3 Algebraic High-Level Nets

In this section we review the concept of algebraic high-level nets in the notation with typed places as presented in [EHP+02]. Algebraic high-level nets are an integration of classical Petri nets (see Section 2.1) and algebraic specifications (see Section 2.2). Technically, the algebraic specification is used to define net inscriptions by terms over the specification. Furthermore firing conditions defined by algebraic equations guarantee that certain constraints are respected. The marking of algebraic high-level nets consists not only of black tokens, but also of data tokens which are elements from a given algebra. In contrast to black tokens of low level Petri nets data tokens can be modified during the firing of transitions. We employ algebraic high-level net morphisms, i.e. structure compatible mappings between nets, leading to the category of algebraic high-level nets. Here, we use the general notion of morphisms, which act on the specification part as well as on the algebra part of the net.

**Definition 2.3.1 (Algebraic High-Level Net)**
An algebraic high-level (AHL) net $N = (\mathrm{SP}, P, T, pre, post, cond, type, A)$ consists of

- an algebraic specification $\mathrm{SP} = ((S, OP, Y), E)$ (see Def. 2.2.12) and additional variables $X$ such that $\Sigma = (S, OP, X)$ is a signature with variables (see Def. 2.2.2),
- a set of places $P$ and a set of transitions $T$,
- pre- and post domain functions $pre, post : T \longrightarrow (T_\Sigma(X) \otimes P)^\oplus$,
- firing condition function $cond : T \longrightarrow \mathcal{P}_{fin}(Eqns(\Sigma))$,
- a type function $type : P \longrightarrow S$, and
- a $SP$-algebra $A$,

where $(T_\Sigma(X) \otimes P) = \{(term, p) | term \in T_\Sigma(X)_{type(p)}, p \in P\}$ and $Eqns(\Sigma)$ are all equations over the signature $\Sigma$ (see Def. 2.2.11).

**Definition 2.3.2 (Firing Behavior of AHL-Nets)**
Given an AHL-net $N = (\mathrm{SP}, P, T, pre, post, cond, type, A)$, then

- a marking of $N$ is given by $M \in CP^\oplus$ where

$$CP = (A \otimes P) = \{(a, p) | a \in A_{type(p)}, p \in P\};$$

- the set of variables $Var(t) \subseteq X$ of a transition $t \in T$ are the variables of the net inscriptions in $pre(t), post(t)$, and $cond(t)$. Let $v : Var(t) \longrightarrow A$ be a variable valuation with term evaluation $v^\sharp : T_\Sigma(Var(t)) \longrightarrow A$, then $(t, v)$ is a consistent transition valuation iff $cond(t)$ is validated in $A$ under $v$ (see Def. 2.2.11). The set $CT$ of consistent transition valuations is defined by

$$CT = \{(t, v) | t \in T, v : Var(t) \longrightarrow A, (t, v) \text{ consistent transition valuation}\};$$

- a transition $t \in T$ is enabled in $M$ under $v$ iff $(t, v) \in CT$ and $pre_A(t, v) \leq M$, where $pre_A : CT \longrightarrow CP^\oplus$ is defined by $pre_A(t, v) = \hat{v}(pre(t)) \in (A \otimes P)^\oplus$ and $\hat{v} : (T_\Sigma(Var(t)) \otimes P)^\oplus \longrightarrow (A \otimes P)^\oplus$ is the obvious extension of $v^\sharp$ to terms and places (similar $post_A : CT \longrightarrow CP^\oplus$);

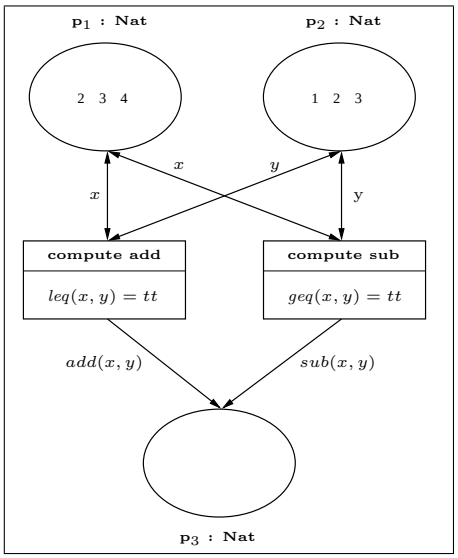



Figure 2.2: Algebraic high-level net "Computation"

- if $t \in T$ is enabled in $M$ under $v$ the follower marking $M'$ is computed by

$$M' = M \ominus pre_A(t, v) \oplus post_A(t, v)$$

and denoted by $M[(t, v)\rangle M'$.

**Example 2.3.3 ("Computation" as Algebraic High-Level Net)**
Fig. 2.2 shows a very simple algebraic high-level net. The idea of the net is to compute alternatively the addition or the subtraction of natural numbers. The net is inscribed with terms over the classical signature NAT of natural numbers and truth values given below. In fact, it is sufficient to consider as specific specification a signature, i.e. a specification with empty set of equations, because we define an explicit NAT-algebra instead of using the initial model. The set of operations of the signature NAT consists exactly of those operations which are denoted in the net inscription of the algebraic high-level net in Fig. 2.2.

$$\begin{array}{ll} \text{NAT} = & \\ \text{sorts:} & Bool, Nat \\ \text{opns:} & tt,\ ff :\rightarrow Bool \\ & leq, geq : Nat\ Nat \rightarrow Bool \\ & add, sub : Nat\ Nat \rightarrow Nat \end{array}$$

We consider the NAT-algebra $A$ with the carrier $A_{Nat}$ consisting of natural numbers and an error element, i.e. $A_{Nat} = \mathbb{N} \cup \{undef\}$, and the carrier $A_{Bool}$ consisting of truth values, i.e. $A_{Bool} = \{true, false\}$. Moreover, we have the constants $tt_A = true$, $ff_A = false$ and the well-known functions $leq_A, geq_A, add_A$, and $sub_A$, respectively, where $sub_A(n_1, n_2) = undef$ if $n_1 \leq n_2$ and $add(n_1, n_2) = sub_A(n_1, n_2) = undef$ if $n_1 = undef$ or $n_2 = undef$. Finally, the application of the error element within the functions $leq_A$ and $geq_A$ results in the truth value $false$.

In Figure 2.2 the type of places $p_1$ and $p_2$ is given by the sort $Nat$. The initial marking of place $p_1$ consists of $2, 3, 4 \in \mathbb{N}$ and the initial marking of place $p_2$ consists of $1, 2, 3 \in \mathbb{N}$, i.e. $M = \sum_{i=2}^{4}(i, p_1) \oplus \sum_{j=1}^{3}(j, p_2)$.

To demonstrate the firing behavior of the transition *compute sub* (and similar for *compute add*) we first have to give a variable valuation $v$ to the variables $x$ and

$y$ defined by $v(x) = 3$ and $v(y) = 2$. Because the firing condition $geq(x, y) = tt$ is validated in $A$ under $v$, the follower marking is computed as follows: the data element 3 on place $p_1$ and the data element 2 on place $p_2$ remain to be unchanged as indicated by the double arrow, while the result $sub_A(3, 2) = 1$ is added to the place $p_3$.

**Definition 2.3.4 (Algebraic High-Level Net Morphisms)**
Given AHL-nets $N_i = (\mathrm{SP}_i, P_i, T_i, pre_i, post_i, cond_i, type_i, A_i)$ for $i \in \{1, 2\}$, an AHL-net morphism $f : N_1 \longrightarrow N_2$ is given by $f = (f_{\mathrm{SP}}, f_P, f_T, f_A)$ with

- a specification morphism $f_{\mathrm{SP}} : \mathrm{SP}_1 \longrightarrow \mathrm{SP}_2$ (see Def. 2.2.14) with signature morphism $f_\Sigma = (f_S, f_{OP}, f_X) : \Sigma_1 \longrightarrow \Sigma_2$ and extension $f_\Sigma^\sharp$ to terms and equations such that the restrictions $f^\sharp_{\Sigma|Var(t)} : Var(t) \longrightarrow Var(f_T(t))$ of $f_\Sigma$ to variables of transitions $t \in T_1$ are bijective,
- a function $f_P : P_1 \longrightarrow P_2$,
- a function $f_T : T_1 \longrightarrow T_2$, and
- $f_A : A_1 \longrightarrow A_2$ is induced by an isomorphism $f_a : A_1 \longrightarrow V_{f_\Sigma}(A_2)$ in $\mathbf{Alg(SP}_1\mathbf{)}$ where $V_{f_\Sigma}$ is a forgetful functor (see Def. 2.2.16)

such that the following diagrams commute componentwise:

$$\begin{array}{ccccc}
\mathcal{P}_{fin}(Eqns(\Sigma_1)) & \xleftarrow{cond_1} & T_1 & \underset{post_1}{\overset{pre_1}{\rightrightarrows}} & (T_{\Sigma_1}(X_1) \otimes P_1)^\oplus \\
\Big\downarrow{\scriptstyle \mathcal{P}_{fin}(f_\Sigma^\sharp)} & = & \Big\downarrow{\scriptstyle f_T} & = & \Big\downarrow{\scriptstyle (f_\Sigma^\sharp \otimes f_P)^\oplus} \\
\mathcal{P}_{fin}(Eqns(\Sigma_2)) & \xleftarrow[cond_2]{} & T_2 & \underset{post_2}{\overset{pre_2}{\rightrightarrows}} & (T_{\Sigma_2}(X_2) \otimes P_2)^\oplus
\end{array}$$

$$\begin{array}{ccc}
P_1 & \xrightarrow{type_1} & S_1 \\
\Big\downarrow{\scriptstyle f_P} & = & \Big\downarrow{\scriptstyle f_S} \\
P_2 & \xrightarrow[type_2]{} & S_2
\end{array}$$

where $(f_\Sigma^\sharp \otimes f_P)$ denotes the corresponding function of type consisting arc inscriptions defined for all $(term, p) \in (T_{\Sigma_1}(X_1) \otimes P_1)$ by

$$(f_\Sigma^\sharp \otimes f_P)(term, p) = (f_\Sigma^\sharp(term), f_P(p))$$

and $(f_\Sigma^\sharp \otimes f_P)^\oplus$ is the unique homomorphic extension of $(f_\Sigma^\sharp \otimes f_P)$. $\mathcal{P}_{fin}(f_\Sigma^\sharp)$ is the extension of $f_\Sigma$ to powersets, i.e. for $\{e_1, \ldots, e_n\} \in \mathcal{P}_{fin}(Eqns(\Sigma_1)), n \geq 0$, we have

$$\begin{array}{l}
\mathcal{P}_{fin}(f_\Sigma^\sharp)(\emptyset) = \emptyset \text{ and} \\
\mathcal{P}_{fin}(f_\Sigma^\sharp)(\{e_1, \ldots, e_n\}) = \{f_\Sigma^\sharp(e_1), \ldots, f_\Sigma^\sharp(e_n)\} \text{ for } n \geq 1.
\end{array}$$

**Remark 2.3.5 (Restriction of Variables of Transitions)**
The restriction $f^\sharp_{\Sigma|Var(t)} : Var(t) \longrightarrow Var(f_T(t))$ of $f_\Sigma$ to variables of transitions $t \in T_1$ have to be bijective for the preservation of the firing behavior by AHL-net morphisms.

**Definition 2.3.6 (Category AHLNet)**
The category **AHLNet** consists of AHL-nets as objects and of AHL-net morphisms as morphisms, where the composition is defined componentwise.

## 2.4 Example: House of Philosophers as AHL-Net

In order to illustrate the concepts described above we will present a small system inspired by the case study "the Hurried Philosophers" of C. Sibertin-Blanc proposed in [SB01] which is a refinement of the well-known classical "Dining Philosophers". According to the requirements of the hurried philosophers in [SB01] the philosophers have the capability to introduce a new guest at the table, which - in the case of low level Petri nets - certainly changes the net structure of the token net representing the philosophers at the table. The intention is to consider the change of the net structure as rule-based transformation of Petri nets in the sense of graph transformation systems [Ehr79, Roz97]. In order to integrate the token game of Petri nets with rule-based transformations, we propose in this section the new paradigm "nets and rules as tokens". Of course, this concept has interesting applications in all areas where dynamic changes of the net structure have to be considered while the system is still running.

### 2.4.1 Requirements

In our case study "House of Philosophers" presented below we essentially consider the following requirements:

1. There are three different locations in the house where the philosophers can stay: the library, the entrance-hall and the restaurant;
2. In the restaurant there are different tables where one or more philosophers can be placed to have dinner;
3. Each philosopher can eat at a table only when he has both forks, i.e. the philosophers at each table follow the rules of the classical "Dining Philosophers";
4. The philosophers in the entrance-hall have the following additional capabilities:
   (a) They are able to invite another philosopher in the entrance-hall to enter the restaurant and to take place at one of the tables;
   (b) They are able to ask a philosopher at one of the tables with at least two philosophers to leave the table and to enter the entrance-hall.

### 2.4.2 Data Type Part

In order to allow P/T-systems and rules as tokens of an AHL-net we provide a specific specification SYSTEM-SIG and SYSTEM-SIG-algebra $A$ based on the definitions and constructions presented in Section 2.1. In fact, for our example it is sufficient to consider a specific signature instead of a specification, i.e. the set of equations is empty. Given vocabularies $T_0$ and $P_0$, the signature SYSTEM-SIG is given by

SYSTEM-SIG =
sorts: $Transitions, Places, Bool, System, Mor, Rules$
opns: $tt, ff :\rightarrow Bool$
$enabled : System \times Transitions \rightarrow Bool$
$fire : System \times Transitions \rightarrow System$
$applicable : Rules \times Mor \rightarrow Bool$
$transform : Rules \times Mor \rightarrow System$
$coproduct : System \times System \rightarrow System$
$isomorphic : System \times System \rightarrow Bool$
$cod : Mor \rightarrow System$

and the SYSTEM-SIG-algebra $A$ for P/T-systems and rules is given by

- $A_{Transitions} = T_0, A_{Places} = P_0, A_{Bool} = \{true, false\}$,
- $A_{System}$ the set of all P/T-systems over $T_0$ and $P_0$, i.e.
  $A_{System} = \{PN | PN = (P, T, pre, post, M)$ P/T-system, $P \subseteq P_0, T \subseteq T_0\}$
  $\cup \{undef\}$,
- $A_{Mor}$ the set of all P/T-system morphisms for $A_{System}$, i.e.
  $A_{Mor} = \{f | f : PN \longrightarrow PN'$ P/T-system morphism
  with $PN, PN' \in A_{System}\}$,
- $A_{Rules}$ the set of all rules of P/T-systems, i.e.
  $A_{Rules} = \{r | r = (L \xleftarrow{i_1} I \xrightarrow{i_2} R)$ rule of P/T-systems with
  strict inclusions $i_1, i_2\}$,
- $tt_A = true$, $\mathit{ff}_A = false$,
- $enabled_A : A_{System} \times T_0 \longrightarrow \{true, false\}$ for $PN = (P, T, pre, post, M)$ with

$$enabled_A(PN, t) = \begin{cases} true & \text{if } t \in T, pre(t) \leq M \\ false & \text{else} \end{cases}$$

- $\mathit{fire}_A : A_{System} \times T_0 \longrightarrow A_{System}$ for $PN = (P, T, pre, post, M)$ with

$$\mathit{fire}_A(PN, t) = \begin{cases} (P, T, pre, post, M \ominus pre(t) \oplus post(t)) & \\ & \text{if } enabled_A(PN, t) = tt \\ undef & \text{else} \end{cases}$$

- $applicable_A : A_{Rules} \times A_{Mor} \longrightarrow \{true, false\}$ with

$$applicable_A(r, m) = \begin{cases} true & \text{if } r \text{ is applicable at match } m \\ false & \text{else} \end{cases}$$

- $transform_A : A_{Rules} \times A_{Mor} \longrightarrow A_{System}$ with

$$transform_A(r, m) = \begin{cases} H & \text{if } applicable_A(r, m) \\ undef & \text{else} \end{cases}$$

  where for $L \xrightarrow{m} G$ and $applicable_A(r, m) = true$ we have a direct transformation $G \xRightarrow{r} H$,

- $coproduct_A : A_{System} \times A_{System} \longrightarrow A_{System}$ the disjoint union (i.e. the two P/T-systems are combined without interaction) with

$$coproduct_A(PN_1, PN_2) = \begin{cases} undef & \text{if } PN_1 = undef \vee PN_2 = undef \\ ((P_1 \uplus P_2), (T_1 \uplus T_2), pre_3, post_3, M_1 \oplus M_2) & \\ & \text{else} \end{cases}$$

  where $pre_3, post_3 : (T_1 \uplus T_2) \to (P_1 \uplus P_2)^{\oplus}$ are defined by

$$pre_3(t) = \underline{\text{if}}\ t \in T_1\ \underline{\text{then}}\ pre_1(t)\ \underline{\text{else}}\ pre_2(t),$$
$$post_3(t) = \underline{\text{if}}\ t \in T_1\ \underline{\text{then}}\ post_1(t)\ \underline{\text{else}}\ post_2(t).$$

- $isomorphic_A : A_{System} \times A_{System} \longrightarrow \{true, false\}$ with

$$isomorphic_A(PN_1, PN_2) = \begin{cases} true & \text{if } PN_1 \cong PN_2 \\ false & \text{else} \end{cases}$$

  where $PN_1 \cong PN_2$ means that there is a strict P/T-morphism $f = (f_P, f_T)$ with $f : PN_1 \longrightarrow PN_2$ such that $f_P, f_T$ are bijective functions,

- $cod_A : A_{Mor} \longrightarrow A_{System}$ with $cod_A\ (f : PN_1 \longrightarrow PN_2) = PN_2$.

### 2.4.3 System Level

In Fig. 2.3 we present the system level of our version of the case study. The system level is given by an AHL-net, which is explained in Section 2.3. The marking of the AHL-net shows the distribution of the philosophers at different places in the house; the firing behavior of the AHL-net describes the mobility of the philosophers. There are three different locations in the house where the philosophers can stay: the library, the entrance-hall, and the restaurant. Each location is represented by its own place in the AHL-net in Fig. 2.3. Initially there are two philosophers in the library, one philosopher in the entrance-hall, and four additional philosophers are at table 1 resp. table 2 (see Fig. 2.6 and 2.7) in the restaurant.

Philosophers may move around, which means they might leave and enter the library and they might leave and enter the tables in the restaurant. The mobility aspect of the philosophers is modeled by transitions termed *enter* and *leave library* as well as *enter* and *leave restaurant* in our AHL-net in Fig. 2.3. While the philosophers are moving around, the static structure of the philosophers is changed by rule-based transformations. For instance a philosopher enters the restaurant and arrives at a table. Then the structure and the seating arrangement of the philosophers have to be changed. For this reason, we have tokens of type *Rules*, $rule_1$, ..., $rule_4$, which are used as resources. Because the philosophers have their own internal behavior, there are two transitions, *start/stop reading* and *start/stop activities*, to realize the change of the behavior.

### 2.4.4 Token Level

The token level consists of two different types of tokens: P/T-systems and rules. They are represented as tokens in the places typed *System* and *Rules* of the AHL-net in Fig. 2.3. The tokens on system places are modeled by P/T-systems, i.e. Petri nets with an initial marking. In Fig. 2.4 the net $phi_1$ of philosopher 1 is depicted, which - in the state *thinking* - is used as a token on the place *Library* in Fig. 2.3. To start reading, we use the transition *start/stop reading* of the AHL-net in Fig. 2.3. First the variable $n$ is assigned to the net $phi_1$ of the philosopher 1 and the variable $t$ to a transition $t_0 \in T_0$ where $T_0$ is a given vocabulary of transitions. The condition *enabled(n,t)=tt* means that under this assignment $t_0$ is an enabled transition in the net $phi_1$. The evaluation of the term $fire(n, t)$ computes the follower marking of the net (i.e. token $reading_1$) and we obtain the new net $phi'_1$ of the philosopher 1 depicted in Fig. 2.4.

### 2.4.5 Mobility of philosophers by application of rules

We assume that the philosopher 1 wants to leave the library, i.e. the transition *leave library* in the AHL-net in Fig. 2.3 must fire. For this purpose we have to give an assignment for the variables $n, r$ and $m$ in the net inscriptions of the transition. They are assigned to the net $phi_1$ (see Fig. 2.4), the rule $rule_1$ (see Fig. 2.5), and a match morphism $m_1 : L' \longrightarrow G$ between P/T-systems. The first condition *cod*

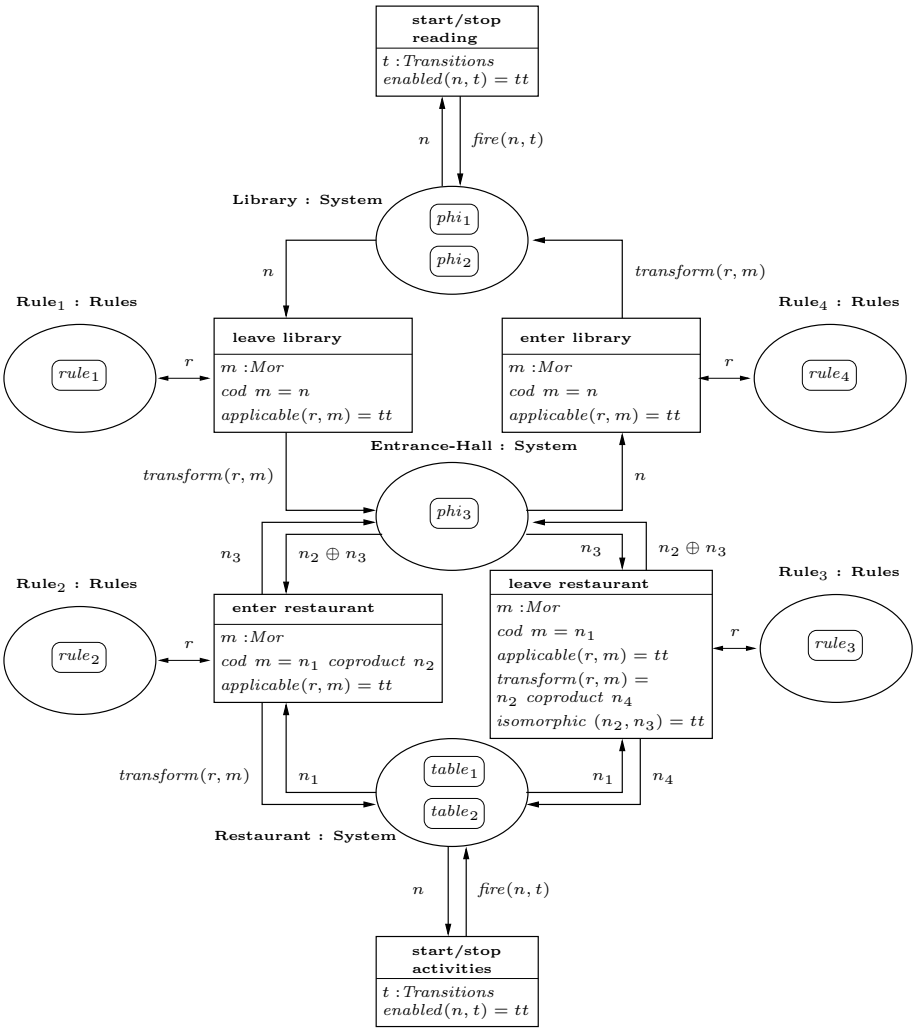



Figure 2.3: Algebraic high-level net of "House of Philosophers"

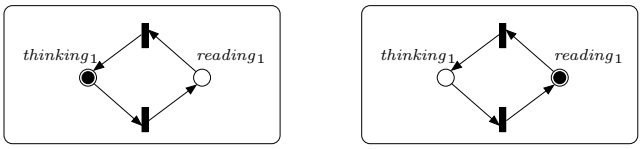



Figure 2.4: Token nets $phi_1$ and $phi'_1$ of philosopher 1

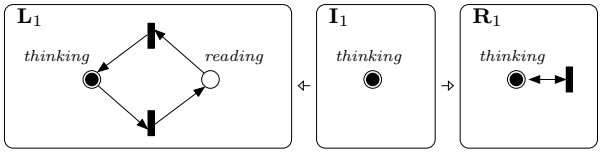



Figure 2.5: Token rule $rule_1$

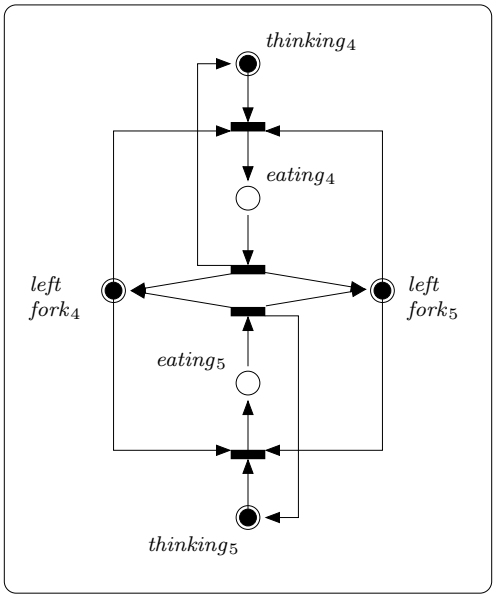



Figure 2.6: Token net $table_1$ of philosopher 4 and 5 at table 1

$m$=$n$ requires $G = phi_1$ and the second condition *applicable(r,m)=tt* makes sure that rule $rule_1$ is applicable to $phi_1$, especially $L' = L_1$, s.t. the evaluation of the term *transform(r,m)* leads to the new net $phi_1''$ isomorphic to $R_1$ of $rule_1$ in Fig. 2.5. As result of this firing step $phi_1$ is removed from place *Library* and $phi_1''$ is added on place *Entrance-Hall*.

In a further step the philosopher 1 is invited by the philosopher 3 to enter the restaurant in order to take place as a new guest at the table 1. The philosopher 3 accompanies philosopher 1 but returns to the entrance-hall. The token net $phi_3$ of philosopher 3 is isomorphic to $R_1$ of $rule_1$ in Fig. 2.5 where *thinking* in $R_1$ is replaced by $thinking_3$. Currently the philosophers 4 and 5 are at the table 1 (see Fig. 2.6). Both philosophers may start eating, but apparently compete for their shared forks, where *left* $fork_4$=*right* $fork_5$ and *left* $fork_5$=*right* $fork_4$. Analogously table 2 has the same net structure as table 1 but different philosophers are sitting at table 2 (see Fig. 2.7). To introduce the philosopher 1 at the table 1 the seating arrangement at table 1 has to be changed. In our case the new guest takes place between philosopher 4 and 5. Formally, we apply rule $rule_2 = (L_2 \xleftarrow{i_1} I_2 \xrightarrow{i_2} R_2)$, which is depicted in the upper row of Fig. 2.8 and used as token on place $Rule_2$. We have to give an assignment $v$ for the variables of the transition *enter restaurant*, i.e. variables $n_1, n_2, n_3, r$, and $m$. The assignment $v$ is defined by $v(n_1) = table_1$, $v(n_2) = phi_1''$, $v(n_3) = phi_3$, $v(r) = rule_2$, and $v(m) = g$ (see match morphism $g : L_2 \longrightarrow G$ in Fig. 2.8). Then we compute the disjoint union of the P/T-system $phi_1''$ and the P/T-system $table_1$ as denoted by the net inscription $n_1$ *coproduct* $n_2$ in the firing condition of the transition *enter restaurant*. The result is the disjoint union of both nets shown as P/T-system $G$ in Fig. 2.8.

In our case the match $g$ maps $thinking_j$ and $eating_j$ in $L_2$ to $thinking_4$ and $eating_4$ in $G$ of Fig. 2.8. The condition *cod* $m = n_1$ *coproduct* $n_2$ makes sure that the codomain of $g$ is equal to $G$. The second condition *applicable(r,m)=tt* checks if $rule_2$ is applicable with match $g$ to $G$ (see "gluing condition" (Def. 2.1.15) and "applicability" (Def. 2.1.18) in Section 2.1). In the direct transformation shown in

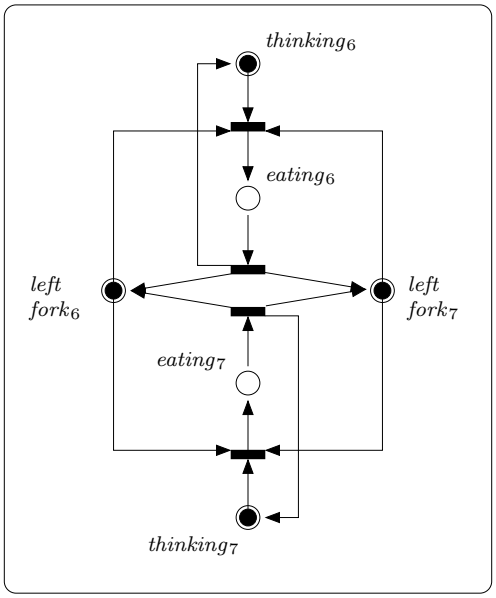



Figure 2.7: Token net $table_2$ of philosopher 6 and 7 at table 2

Fig. 2.8 we delete in a first step $g(L_2 \setminus I_2)$ from $G$ leading to P/T-system $C$. Note that a positive check of the "gluing condition" makes sure that $C$ is a well-defined P/T-system (see Fact 2.1.16 in Section 2.1). In a second step we glue together the P/T-systems $C$ and $R_2$ along $I_2$ leading to P/T-system $H$ in Fig. 2.8. $H$ shows the new version of table 1 given by the net $table'_1$ of table 1, where philosophers 1, 4 and 5 are sitting at the table, all of them in state *thinking*. The effect of firing the transition *enter restaurant* in Fig. 2.3 with assignments of variables as discussed above is the removal of P/T-systems $phi''_1$ from place *Entrance Hall* and $table_1$ from place *Restaurant* and adding P/T-System $table'_1$ to the place *Restaurant*.

Philosophers in the entrance-hall have the capability to ask one of the philosophers in the restaurant to leave; this is realized in our system by the transition *leave restaurant* in Fig. 2.3. We use the rule $rule_3$ defined as inverse of $rule_2$ in Fig. 2.8, i.e. $rule_3 = (R_2 \xleftarrow{i_2} I_2 \xrightarrow{i_1} L_2)$, which is present as a token on place $Rule_3$. This rule is applied with inverse direct transformation to the one depicted in Fig. 2.8. Finally, the rule $rule_4$ is the inverse of rule $rule_1$ (see Fig. 2.5) enabling the philosopher to enter the library by firing of the transition *enter library* in Fig. 2.3. We have to guarantee that after the application of $rule_3$ the philosopher who is leaving the restaurant goes into the entrance-hall. In our case one philosopher is asked by philosopher 3 in the entrance-hall to leave the table. Formally this is denoted by the firing condition $isomorphic(n_2, n_3) = tt$ which ensures that the net of the philosophers denoted by $n_2$ is isomorphic to the net $phi_3$ of philosopher 3 denoted by $n_3$.

The execution of philosopher activities at different tables, i.e. the firing of the transition *start/stop activities* in Fig. 2.3, is analogously defined as the firing of the transition *start/stop reading* described above.

### 2.4.6 Validation of Requirements

Our case study "House of Philosophers" satisfies the requirements presented in the beginning of this section.

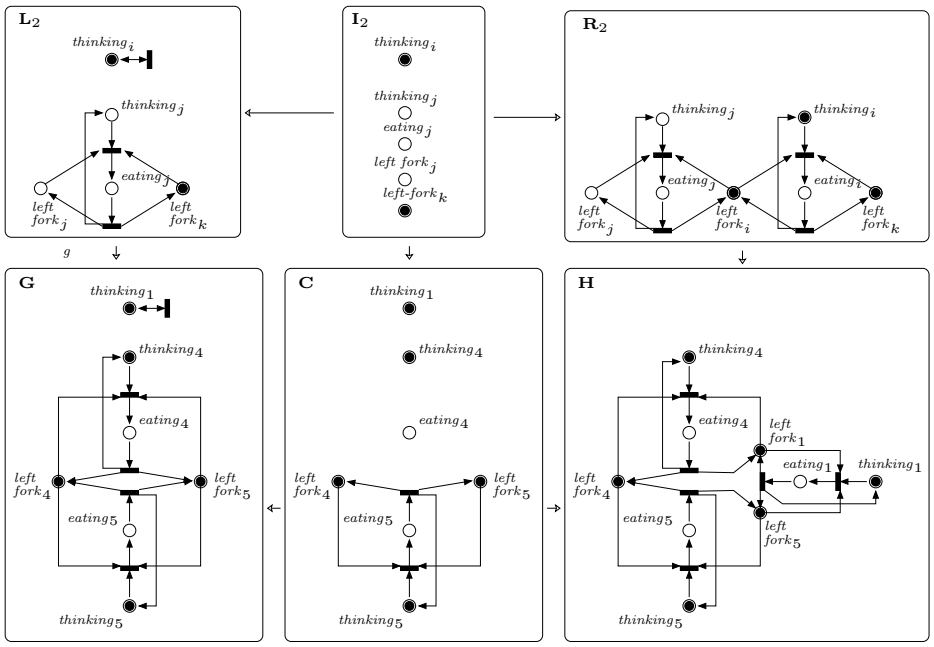



Figure 2.8: Direct Transformation

1. The three different locations in the house are represented by places *Library*, *Entrance-Hall*, and *Restaurant* in Fig. 2.3;

2. In the initial state we have the two tables $table_1$ with philosophers 4 and 5 and $table_2$ with philosophers 6 and 7 on place *Restaurant*. In a later state also philosopher 1 is sitting at table 1 as shown by net $H$ of Fig. 2.8;

3. If there are $n \geq 2$ philosophers sitting at each table, the table with $n$ philosophers is presented by the classical "Dining Philosophers" net;

4. The capability of a philosopher in the entrance-hall to invite another philosopher to enter (leave) the restaurant is given by firing of the transition *enter restaurant (leave restaurant)* in Fig. 2.3. The applicability of the rule $rule_3$ ensures that a philosopher only leaves a table with at least two philosophers.

# Chapter 3

# Algebraic Higher-Order Nets

In this chapter we formally introduce our new concept of algebraic higher-order nets as an extension of algebraic high-level nets, presented in the previous chapter, by higher-order features and partiality. The most prominent new aspect is that the data type part of algebraic higher-order nets is given by higher-order algebras instead of classical algebras. This results in a great impact on the structure of algebraic higher-order nets. On the theoretical level the higher-order type structure of higher-order algebras is transferred to algebraic higher-order nets. Thus, we are able to distinguish between different higher-order types of places like function types and product types. Because in our concept functions are first-class citizens, we get a natural way of using functions as tokens and the powerful and elegant concept of higher-order functions can be used in the net inscriptions. Moreover, we achieve the notion of atomic formulas to define firing conditions of algebraic higher-order nets. From a practical point of view our models are on a high-level of abstraction and support structural flexibility and system adaptability in an extensive way. In our approach the order of functions is allowed to be arbitrarily high. So the basic idea is to reflect a group of functions as tokens, which may be dynamically combined at run-time. This mechanism is called operation late-binding in our context. A token is then recursively defined as a function and a function can shrink to a token.

In Section 3.1 we formally investigate higher-order algebras. Some of the notions and results have already been published in [Wol05] but are revised and extended for our approach. The main results achieved in this thesis for higher-order algebras concern the finite cocompleteness of the category of higher-order signatures and the functorial construction of higher-order terms for each higher-order signature. These results are essential to ensure the finite cocompleteness of the category of algebraic higher-order net schemes in the subsequent chapter. Based on these definitions we give the necessary notions of algebraic higher-order net schemes and algebraic higher-order nets and formulate the operational behavior of algebraic higher-order nets in Section 3.2. In order to illustrate the practical use of algebraic higher-order nets in Section 3.3, we present a higher-order version of the simple example "Computation" (see Section 2.3). Within this example we also give a rough idea of the main results which will be obtained in the subsequent chapters.

## 3.1 Higher-Order Partial Algebras

In this section we present an approach of higher-order partial algebras developed in [Wol05]. This approach was intended to provide an algebraic semantics of functional languages but turned out to be well suited for algebraic higher-order nets. Higher-order algebras are staying close to the set-theoretic semantics of classical algebras

as presented in Section 2.2. In fact, there is an embedding of classical signatures into higher-order signature which will be discussed later in Section 10.2. In the following we discuss the main items of higher-order algebras in more detail, which differ from classical algebras and are essential for algebraic higher-order nets.

**Partiality** There is a practical need for a systematic treatment of partial operations to handle errors and exceptions, and an account for non-terminating operations [Fef92]. Partial operations are useful to represent functions which are not yet specified during the design process and may not yield a result on some possible input, as for instance predecessor for positive natural numbers, head and tail for lists, or pop and top for stacks. Another relevant example arises due to the machine limits, i.e. data types are limited, such that the constructors of infinite data type systems like push for stacks are not defined on the extreme values and hence become partial. In a total setting these operations require an error management mechanism, i.e. partial functions are simulated by means of total functions on sorts which are equipped with one or more error elements. But having added (at least) one new element, the application of operations to it has to be specified as well and the size of specifications dramatically increases. A further practical drawback is that the insertion of error elements is a typical example of non-persistent parameterized specification, thus there is no passing compatibility nor actual parameter protection (see [EM85]). Moreover, partial recursive functions with non recursive domain are needed for example to specify the interpreter of a programming language, but it is undecidable whether the interpreter yields an output or not and therefore there is obviously no possibility to detect the errors by its application. While in a total setting there does not exist a specification, the specification of partial recursive functions with non recursive domain is possible within the error algebra approach [GTW78]; in semicomputable models, however, it leads to infinite amounts of error values. Another problem arises when the implementation has another version of explicit error handling. Because it becomes difficult to relate the intended model of the specification to the real implementation, the effort for explicit error handling on the level of specifications turns out to be quite useless. Therefore, we introduce a more powerful framework based on the (possibly) partial interpretation of operation symbols to easily describe not only partiality arising from the situations given above, but also semidecidable predicates by specifying the positive kernel of the data types.

**Function Types** Due to higher-order functions, which take functions as parameters and/or yield functions as results, we have the possibility to summarize the behavior of a number of functions in an elegant and abstract way, for instance map and filter for lists. But we have to decide about standard, extensional, and intensional function spaces. In standard models each function type is interpreted as the full function space but due to the well known Gödel incompleteness theorem, there cannot be a recursively axiomatized sound and complete calculus for the standard model semantics. Requiring possibly non-standard, but still extensional function spaces means that function types are interpreted by a subset of the full function space, but destroys the existence of initial models. For the total case, there is a way out by restricting oneself to reachable models [MTW87]. But for the partial case, even simple higher-order signatures do not longer have initial models [AC92]. Thus we prefer the notion of intensional function space, i.e. function types are interpreted by arbitrary sets with an application operation of appropriate type. Intensional algebras behave nicely with regard to completeness and existence of free algebras [Poi86]. The advantage is that we can distinguish between different ways of computing a function. For instance, quicksort and bubblesort for list are equal

from an extensional point of view, but from a practical point of view they may be considered as different as they are implemented by quite different algorithms and it is a well-known fact that quicksort is the efficient one of sorting algorithms for lists.

**Product Types** In classical algebras only single sorts are allowed as codomains of functions. Thus functions into an $n$-ary product have to be encoded into a collection of $n$ functions into single sorts. To make our theory more applicable for the user we introduce product types with projection symbols of appropriate types to allow nested products as codomains of functions.

**Predicates** In the classical case there are two main directions to introduce predicates into signatures. In the first possibility a signature is enriched by a boolean sort and some operations into the boolean sort. But treating predicates as operations have the drawback that the truth as well as the untruth of each relation has to be stated, and in particular semicomputable relations cannot be conditionally axiomatized because their falseness cannot be recursively axiomatized. In the second possibility a signature is enriched by a set of typed predicate names, and algebras are modified to include relations on their carriers to interpret predicate names. In the higher-order case with product types and partiality predicates are fully determined by partial functions into a singleton set, i.e. the domain of definition reflects the trueness of the predicate. As we will discuss in Section 7.1, this formalism provides initial/free semantics.

We first introduce a classical specification of higher-order typing structures instead of a direct definition of higher-order types. So we achieve the result that higher-order types are freely generated by a set of basic types. The advantage is that we can use the notions and results introduced in Section 2.2.

**Definition 3.1.1 (Higher-Order Typing Structures)**
Let a (classical) specification $\mathrm{HOTS} = (\Sigma, E)$ be given with

$$
\begin{array}{ll}
\mathrm{HOTS} = & \\
\text{sorts:} & s \\
\text{opns:} & \mathit{unit} :\rightarrow s \\
 & \rightarrow : s\ s \rightarrow s \\
 & \star : s\ s \rightarrow s \\
\text{var:} & \mathit{type}, \mathit{type}_1, \mathit{type}_2, \mathit{type}_3 : s \\
\text{eqns:} & (\mathit{type}_1 \star \mathit{type}_2) \star \mathit{type}_3 = \mathit{type}_1 \star (\mathit{type}_2 \star \mathit{type}_3) \\
 & \mathit{type} \star \mathit{unit} = \mathit{type} \\
 & \mathit{unit} \star \mathit{type} = \mathit{type}
\end{array}
$$

A higher-order typing structure is a HOTS-algebra $(A, \mathit{unit}_A, \rightarrow_A, \star_A)$. If no confusion arises, we omit the index $A$. Given two higher-order typing structures $(A_i, \mathit{unit}, \rightarrow, \star)$ for $i \in \{1, 2\}$. Then a higher-order typing structure morphism

$$f_A : (A_1, \mathit{unit}, \rightarrow, \star) \longrightarrow (A_2, \mathit{unit}, \rightarrow, \star)$$

is a homomorphism of (classical) algebras, i.e. for $\mathit{type}_1, \mathit{type}_2 \in A_1$ we have

$$
\begin{array}{l}
f_{A,s}(\mathit{unit}) = \mathit{unit}, \\
f_{A,s}(\mathit{type}_1 \rightarrow \mathit{type}_2) = f_{A,s}(\mathit{type}_1) \rightarrow f_{A,s}(\mathit{type}_2), \text{ and} \\
f_{A,s}(\mathit{type}_1 \star \mathit{type}_2) = f_{A,s}(\mathit{type}_1) \star f_{A,s}(\mathit{type}_2).
\end{array}
$$

**Definition 3.1.2 (Category Alg(HOTS))**
The category **Alg(HOTS)** consists of higher-order typing structures $(A, \mathit{unit}, \rightarrow, \star)$ as objects and higher-order typing structure morphisms as morphisms.

**Definition 3.1.3 (Forgetful and Free Functors)**
The forgetful functor $V_{f_{\text{HOTS}}}$ : **Alg(HOTS)** $\longrightarrow$ **Sets** is defined for all higher-order typing structures by

$$V_{f_{\text{HOTS}}}(A, unit, \rightarrow, \star) = A$$

and for all higher-order typing structure morphism

$$f_A : (A_1, unit, \rightarrow, \star) \longrightarrow (A_2, unit, \rightarrow, \star)$$

by

$$V_{f_{\text{HOTS}}}(f_A) = f_A : A_1 \longrightarrow A_2.$$

Moreover, the forgetful functor $V_{f_{\text{HOTS}}}$ : **Alg(HOTS)** $\longrightarrow$ **Sets** has a left adjoint functor $F_{f_{\text{HOTS}}}$ : **Sets** $\longrightarrow$ **Alg(HOTS)**, i.e. there is an adjunction $F_{f_{\text{HOTS}}} \vdash V_{f_{\text{HOTS}}}$.

We get an initial model of higher-order typing structures, i.e. the quotient term algebra. To obtain a special representation we define the set of higher-order types. Here the subset of single higher-order types is important because these types have to be interpreted in the corresponding higher-order partial algebra.

**Definition 3.1.4 (Higher-Order Types)**
For a set $S$ of basic types we define the set $S^{\rightarrow}$ of higher-order types over $S$ and the set $BSF^{\rightarrow}$ of single higher-order types inductively by the following rules:

$$\begin{aligned}
& \Rightarrow && unit \in S^{\rightarrow} \\
& \Rightarrow && S \subseteq BFS^{\rightarrow} \\
type_1, type_2 \in S^{\rightarrow} & \Rightarrow && (type_1 \rightarrow type_2) \in BFS^{\rightarrow} \\
type_1, \ldots, type_n \in BFS^{\rightarrow}, n \geq 2 & \Rightarrow && (type_1 \star \ldots \star type_n) \in S^{\rightarrow} \\
& \Rightarrow && BFS^{\rightarrow} \subseteq S^{\rightarrow}
\end{aligned}$$

where *unit* is called empty type, $(type_1 \rightarrow type_2)$ are called function types and $(type_1 \star \ldots \star type_n)$ are called product types.

Due to Def. 3.1.1 we avoid nested product types. But to simplify our notations, especially in the folding and unfolding constructions in Chapter 5, we introduce the following notation.

**Remark 3.1.5 (Nested Product Types)**
Let $(type_{1,1} \star \ldots \star type_{1,m_1}), \ldots, (type_{n,1} \star \ldots \star type_{1,m_n}) \in S^{\rightarrow}, n, m_1, \ldots, m_n \in \mathbb{N}$. Then we use the notation

$$(type_{1,1} \star \ldots \star type_{1,m_1} \star \ldots \star type_{n,1} \star \ldots \star type_{1,m_n}).$$

**Remark 3.1.6 (Predicates)**
Let $Pred(type)$ abbreviate the function type $type \rightarrow unit$.

Due to the adjoint situation described above we get a Kleisli category, which can be seen as the freely generated higher-order types over a set of basic types. The advantage is that the composition of morphisms are defined by substitution, which will be used in Fact 3.1.13. Note that the category of free commutative monoids as used in Chapter 2 can be also seen as a Kleisli category, where we even have two different kinds of representations, i.e. formal sums and multisets.

**Definition 3.1.7 (Category HOTS)**
The category **HOTS** is the category of freely generated HOTS-algebras over a set of basic types, i.e. the Kleisli category for the adjunction $F_{f_{HOTS}} \vdash V_{f_{HOTS}}$ which takes a set of basic types $S$ to the set of higher-order types $S^{\rightarrow}$. Any function $f : S_1 \longrightarrow S_2^{\rightarrow}$ can be extended to a unique function $f^{\rightarrow} : S_1^{\rightarrow} \longrightarrow S_2^{\rightarrow}$ with $f^{\rightarrow}(s_1) = f(s_1)$ for $s_1 \in S_1$. Moreover, we have

$$(f_2^{\rightarrow} \circ f_1)^{\rightarrow} = f_2^{\rightarrow} \circ f_1^{\rightarrow} \text{ and}$$
$$id_S^{\rightarrow} = id_{S^{\rightarrow}}.$$

**Definition 3.1.8 (Higher-Order Signature)**
A higher-order signature $\Sigma = (S, OP, scr, tar)$ is given by a set $S$ of basic types and a set $OP$ of operation symbols, a source function $src : OP \longrightarrow S^{\rightarrow}$ and a target function $tar : OP \longrightarrow S^{\rightarrow}$. We write $op : type_1 \rightharpoonup type_2$ for $op \in OP$ with $src(op) = type_1$ and $tar(op) = type_2$. $OP$ is required to contain an application symbol

$$apply^{type_1, type_2} : (type_1 \rightarrow type_2) \star type_1 \rightharpoonup type_2$$

for each function type $type_1 \rightarrow type_2 \in S^{\rightarrow}$. Further we assume projection symbols

$$pr_i^{(type_1 \star \ldots \star type_n)} : (type_1 \star \ldots \star type_n) \rightharpoonup type_i$$

with $pr_i^{(type_1 \star \ldots \star type_n)} \notin OP$ for all $(type_1 \star \ldots \star type_n) \in S^{\rightarrow}, 1 \leq i \leq n$, and denote by $OP^{\rightarrow}$ the set consisting of projection symbols and operation symbols.

$$OP^{\rightarrow} = OP \cup \{pr_i^{(type_1 \star \ldots \star type_n)} | (type_1 \star \ldots \star type_n) \in S^{\rightarrow}, 1 \leq i \leq n\}.$$

**Remark 3.1.9 (Constant and Predicate Symbols)**
The notation $c : type \in OP$ is used to indicate that $src(c) = unit$ and denotes a constant symbol and all constant symbols $p : Pred(type)$ are called predicates. An operation symbol $p' : type \rightharpoonup unit \in OP$ is called predicate symbol. Note that $p : Pred(type)$ is an abbreviation of $p : unit \rightharpoonup (type \rightarrow unit)$ and is different from the predicate symbol $p' : type \rightharpoonup unit$.

In algebraic higher-order nets we need a set of variables, which are used in the net inscriptions. So we extend the notion of higher-order signatures by a set of variables. Usually variables are locally bound to some terms. But for our purpose it is more suitable to use a global set of variables.

**Definition 3.1.10 (Higher-Order Signature with Variables)**
Let $\Sigma = (S, OP, scr, tar)$ be a higher-order signature, $X$ a set, called set of variables, with $X \cap OP = \emptyset$ and $var : X \longrightarrow S^{\rightarrow}$ a function, called sorts of variables. Then $\Sigma = (S, OP, scr, tar, X, var)$ is a higher-order signature with variables. We write $x : type$ for $x \in X$ and $var(x) = type$. To simplify the notation, we also denote a higher-order signature with variables by the following three components $(S, OP, X)$.

**Definition 3.1.11 (Higher-Order Signature Morphism)**
Given higher-order signatures $\Sigma_i = (S_i, OP_i, X_i)$ for $i = \{1, 2\}$, a higher-order signature morphism $f_\Sigma : \Sigma_1 \longrightarrow \Sigma_2$ is a tuple of functions

$$f_\Sigma = (f_S : S_1 \longrightarrow S_2, f_{OP} : OP_1 \longrightarrow OP_2, f_X : X_1 \longrightarrow X_2)$$

such that the following diagram commutes componentwise:

$$\begin{array}{ccccc} OP_1 & \overset{src_1}{\underset{tar_1}{\rightrightarrows}} & S_1^{\rightarrow} & \xleftarrow{var_1} & X_1 \\ {\scriptstyle f_{OP}}\downarrow & = & {\scriptstyle f_S^{\rightarrow}}\downarrow & = & \downarrow{\scriptstyle f_X} \\ OP_2 & \overset{src_2}{\underset{tar_2}{\rightrightarrows}} & S_2^{\rightarrow} & \xleftarrow{var_2} & X_1 \end{array}$$

where $f_S^{\rightarrow} : S_1^{\rightarrow} \longrightarrow S_2^{\rightarrow}$ is the unique extension of $f_s : S_1 \longrightarrow S_2^{\rightarrow}$ (see Fact 3.1.7) and $f_s$ is induced by $f_S$, i.e. $f_s(type) = f_S(type)$ for all basic types $type \in S_1$.

**Fact 3.1.12 (Category HOSig)**
The category **HOSig** consists of higher-order signatures with variables as objects and higher-order signature morphisms as morphisms.

**Proof:** We show that the composition defined componentwise is associative and the identity law is satisfied.

**Composition:** Given higher-order signatures $\Sigma_i = (S_i, PP_i, X_i)$ for $i \in \{1, 2, 3\}$ and a pair of higher-order signature morphisms

$$f_{\Sigma,1} = (f_{S,1}, f_{OP,1}, f_{X,1}) : \Sigma_1 \longrightarrow \Sigma_2 \text{ and}$$
$$f_{\Sigma,2} = (f_{S,2}, f_{OP,2}, f_{X,2}) : \Sigma_2 \longrightarrow \Sigma_3.$$

The composition of $f_{\Sigma,1}$ and $f_{\Sigma,2}$ defined componentwise by

$$f_{\Sigma,2} \circ f_{\Sigma,1} = (f_{S,2} \circ f_{S,1}, f_{OP,2} \circ f_{OP,1}, f_{X,2} \circ f_{X,1}) : \Sigma_1 \longrightarrow \Sigma_3$$

is a higher-order signature morphism because functions of basic types, operations resp. variables are closed under composition and

- the source- and target functions are composable:

$$\begin{aligned} f_{S,2}^{\rightarrow} \circ f_{S,1}^{\rightarrow} \circ src_1 &= f_{S,2}^{\rightarrow} \circ src_2 \circ f_{OP,1} \\ &= src_3 \circ f_{OP,2} \circ f_{OP,1} \end{aligned}$$

  (analogously for the target function)

- the sorts of variables are composable:

$$\begin{aligned} f_{S,2}^{\rightarrow} \circ f_{S,1}^{\rightarrow} \circ var_1 &= f_{S,2}^{\rightarrow} \circ var_2 \circ f_{X,1} \\ &= var_3 \circ f_{X,2} \circ f_{X,1}. \end{aligned}$$

**Associativity:** Let $f_{\Sigma,i} = (f_{S,i}, f_{OP,i}, f_{X,i}), i = 1, 2, 3$, be three higher-order signature morphisms. Then the composition is associative because it is defined componentwise, i.e. we have

$$(f_{\Sigma,3} \circ f_{\Sigma,2}) \circ f_{\Sigma,1} = f_{\Sigma,3} \circ (f_{\Sigma,2} \circ f_{\Sigma,1}).$$

**Identity:** The identity is the identity higher-order signature morphisms. Because the composition is defined componentwise, the identity law is satisfied, i.e. we have for $f_\Sigma = (f_S, f_{OP}, f_X) : \Sigma \to \Sigma'$

$$f_\Sigma \circ id_\Sigma = f_\Sigma \text{ and } id_{\Sigma'} \circ f_\Sigma = f_\Sigma.$$

□

Next we show that the category **HOSig** is finitely cocomplete, i.e. it has all finite colimits like initial objects, coequalizer, and pushouts. This result will be used to show the finite cocompleteness of the category of algebraic higher-order net schemes in Section 4.1.

**Fact 3.1.13 (HOSig is finitely cocomplete)**
The category **HOSig** is finitely cocomplete.

**Proof:** We have to show that **HOSig** has an initial object and all pushouts.

**Initial Object** The initial object $\Sigma_\emptyset$ is given by

$$\Sigma_\emptyset = (S_\emptyset, OP_\emptyset, scr_\emptyset, tar_\emptyset, X_\emptyset, var_\emptyset)$$

with

- $S_\emptyset = OP_\emptyset = X_\emptyset = \emptyset$ and
- $scr_\emptyset = tar_\emptyset = var_\emptyset$ the empty function.

Given a higher-order signature $\Sigma = (S, OP, X)$, then the unique higher-order signature morphisms $f_{\Sigma,\emptyset} : \Sigma_\emptyset \longrightarrow \Sigma$ with $f_{\Sigma,\emptyset} = (f_{S,\emptyset}, f_{OP,\emptyset}, f_{X,\emptyset})$ is given by

- $f_{S,\emptyset} = f_{OP,\emptyset} = f_{X,\emptyset}$ the empty function, where the compatibility of the source function, the target function and the sorts of variables holds due to the emptiness of $OP_\emptyset$ and $X_\emptyset$.

**Pushouts** Given higher-order signature $\Sigma_i = (S_i, OP_i, X_i)$ for $i \in \{1, 2, 3\}$ and a pair of higher-order signature morphisms $f_\Sigma = (f_S, f_{OP}, f_X) : \Sigma_1 \longrightarrow \Sigma_2$ and $g_\Sigma = (g_S, g_{OP}, g_X) : \Sigma_1 \longrightarrow \Sigma_3$, then the pushout consisting of the pushout object $\Sigma_4 = (S_4, OP_4, X_4)$ and the pair of higher-order signature morphisms $f'_\Sigma = (f'_S, f'_{OP}, f'_X) : \Sigma_2 \longrightarrow \Sigma_4$ and $g'_\Sigma = (g'_S, g'_{OP}, g'_X) : \Sigma_3 \longrightarrow \Sigma_4$ is constructed by the following pushouts in **Sets**.

$$\begin{array}{ccc} S_1 & \xrightarrow{f_S} & S_2 \\ {\scriptstyle g_S}\downarrow & = & \downarrow{\scriptstyle f'_S} \\ S_3 & \xrightarrow[g'_S]{} & S_4 \end{array} \qquad \begin{array}{ccc} OP_1 & \xrightarrow{f_{OP}} & OP_2 \\ {\scriptstyle g_{OP}}\downarrow & = & \downarrow{\scriptstyle f'_{OP}} \\ OP_3 & \xrightarrow[g'_{OP}]{} & OP_4 \end{array} \qquad \begin{array}{ccc} X_1 & \xrightarrow{f_X} & X_2 \\ {\scriptstyle g_X}\downarrow & = & \downarrow{\scriptstyle f'_X} \\ X_3 & \xrightarrow[g'_X]{} & X_4 \end{array}$$

Moreover, we have due to Fact 3.1.7 the following pushout of higher-order types in **Sets**.

$$\begin{array}{ccc} S_1^{\rightarrow} & \xrightarrow{f_S^{\rightarrow}} & S_2^{\rightarrow} \\ {\scriptstyle g_S^{\rightarrow}}\downarrow & = & \downarrow{\scriptstyle f_S^{\rightarrow\prime}} \\ S_3^{\rightarrow} & \xrightarrow[g_S^{\rightarrow\prime}]{} & S_4^{\rightarrow} \end{array}$$

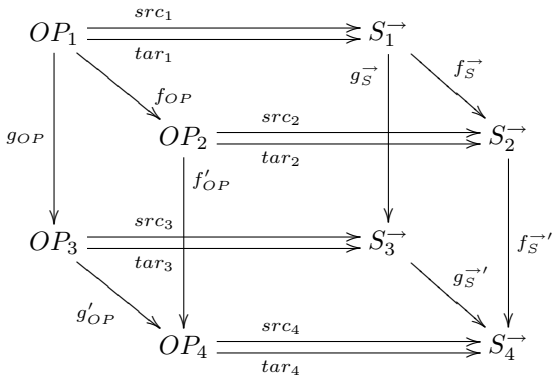
Because $f_\Sigma$ and $g_\Sigma$ are higher-order signature morphisms, we have

$$f_S^{\rightarrow\prime} \circ src_2 \circ f_{OP} = g_S^{\rightarrow\prime} \circ src_3 \circ g_{OP}.$$

Due to the pushout property of $OP_4$ there exists one and only one morphism $src_4 : OP_4 \longrightarrow S_4^{\rightarrow}$ such that

$$f_S^{\rightarrow\prime} \circ src_2 = src_4 \circ f'_{OP} \text{ and } g_S^{\rightarrow\prime} \circ src_3 = src_4 \circ g'_{OP}.$$

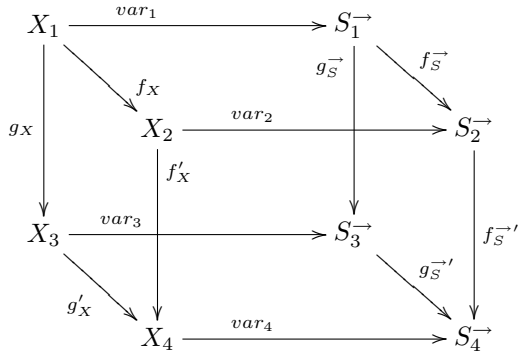
Analogously, the pushout property of $OP_4$ yields $tar_4 : OP_4 \longrightarrow S_4^{\rightarrow}$.




In addition, because $f_\Sigma$ and $g_\Sigma$ are higher-order signature morphisms, we have

$$f_S^{\rightarrow\prime} \circ var_2 \circ f_X = g_S^{\rightarrow\prime} \circ var_3 \circ g_X.$$

Due to the pushout property of $X_4$ there exists one and only one morphism $var_4 : X_4 \longrightarrow S_4^{\rightarrow}$ such that $f_S^{\rightarrow\prime} \circ var_2 = var_4 \circ f'_X$ and $g_S^{\rightarrow\prime} \circ var_3 = var_4 \circ g'_X$.



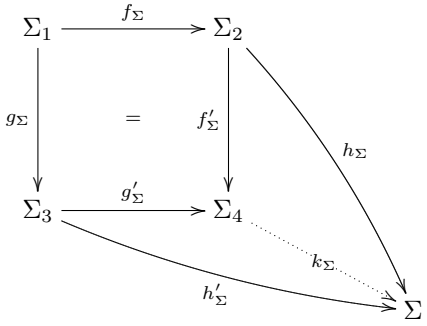

$f'_\Sigma = (f'_S, f'_{OP}, f'_X)$ and $g'_\Sigma = (g'_S, g'_{OP}, g'_X)$ are well-defined, i.e. they satisfy the compatibility of source- and target functions resp. sorts of variables due to the induced morphisms $src_4, tar_4$, and $var_4$. The commutativity follows directly from the componentwise construction of the higher-order signature morphisms and the commutativity of the diagrams above, i.e.

$$\begin{aligned} & (g'_S \circ g_S, g'_{OP} \circ g_{OP}, g'_X \circ g_X) = (f'_S \circ f_S, f'_{OP} \circ f_{OP}, f'_X \circ f_X) \\ \Longrightarrow \quad & g'_\Sigma \circ g_\Sigma = f'_\Sigma \circ f_\Sigma. \end{aligned}$$

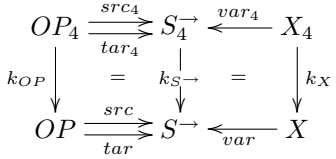
Next we check the pushout property. Given another higher-order signature $\Sigma = (S, OP, X)$ and a pair of higher-order signature morphisms

$$\begin{aligned} h_\Sigma &= (h_S, h_{OP}, h_X) : \Sigma_2 \longrightarrow \Sigma \text{ and} \\ h'_\Sigma &= (h'_S, h'_{OP}, h'_X) : \Sigma_3 \longrightarrow \Sigma \end{aligned}$$

such that $h_\Sigma \circ f_\Sigma = h'_\Sigma \circ g_\Sigma$.




Then the induced morphism $k_\Sigma = (k_S, k_{OP}, k_X) : \Sigma_4 \longrightarrow \Sigma$ is obtained by the induced morphisms $k_S, k_{OP}$, and $k_X$ of each component $S_4, OP_4$ resp. $X_4$. $k$ is well-defined, i.e. it satisfies the compatibility of source- and target functions resp. types of variables due to the induced morphisms $k_S, k_{OP}$ and $k_X$, i.e. the following diagrams commute




where $k_S^{\rightarrow}$ is the unique function induced by $k_S$ (see Fact 3.1.7).

Finally, uniqueness of $k$ follows immediately due to the uniqueness of each component $k_S, k_{OP}$ and $k_X$.

□

In the following we define the set of terms, which are generated by a higher-order signature.

**Definition 3.1.14 (Set of Higher-Order Terms)**
Let $\Sigma = (S, OP, X)$ be a higher-order signature. Then $(T_{\Sigma,type}(X))_{type \in S^{\rightarrow}}$ is the least $S^{\rightarrow}$-sorted set inductively constructed by the following rules:

**1.** $\Rightarrow \langle\rangle \in T_{\Sigma,unit}(X)$

**2.** $x : type \in X \Rightarrow x \in T_{\Sigma,type}(X)$

**3.** $op : type_1 \rightharpoonup type_2 \in OP, term \in T_{\Sigma,type_1}(X)$
$\Rightarrow op(term) \in T_{\Sigma,type_2}(X)$

**4.** $term_1 \in T_{\Sigma,type_1}(X), \ldots, term_n \in T_{\Sigma,type_n}(X), type_i \in BFS^{\rightarrow}$
$\Rightarrow \langle term_1, \ldots, term_n \rangle \in T_{\Sigma,(type_1 \star \ldots \star type_n)}(X)$

**5.** $pr_i^{(type_1 \star \ldots \star type_n)} \in (OP^{\rightarrow} \setminus OP), term \in T_{\Sigma,(type_1 \star \ldots \star type_n)}(X)$
$\Rightarrow pr_i^{(type_1 \star \ldots \star type_n)}(term) \in T_{\Sigma,type_i}(X)$.

The set of $\Sigma$-terms $T_\Sigma(X)$ is defined by $T_\Sigma(X) = \bigcup_{type \in S^{\rightarrow}} T_{\Sigma,type}(X)$.

Instead of using an $S^{\rightarrow}$-sorted set of terms we prefer the notion of a set of terms because in the context of algebraic higher-order nets these terms are used in the net inscriptions. For this reason, we use the notion of the free commutative monoids which are generated over a set of terms (and a set of places).

To form tuples of terms the type of the involved terms have to be of single type because we avoid nested product types (see Def. 3.1.1). But to simplify our notations, especially in the folding and unfolding constructions in Chapter 5, we introduce the following notation.

**Remark 3.1.15 (Nested Product Terms)**
Let $n, m_1, \ldots, m_n \in \mathbb{N}$ and

$$\begin{aligned}\langle term_{1,1}, \ldots, term_{1,m_1}\rangle &\in T_{\Sigma,(type_{1,1}\star\ldots\star type_{1,m_1})}(X)\\ &\vdots\\ \langle term_{n,1}, \ldots, term_{n,m_n}\rangle &\in T_{\Sigma,(type_{n,1}\star\ldots\star type_{n,m_n})}(X).\end{aligned}$$

Then we use the notation

$$\langle term_{1,1}, \ldots, term_{1,m_1}, \ldots, term_{n,1}, \ldots, term_{n,m_n}\rangle$$

for a term of type $(type_{1,1} \star \ldots \star type_{1,m_1} \star \ldots \star type_{n,1} \star \ldots \star type_{n,m_n})$.

**Remark 3.1.16 (Applications and Atomic Formulas)**
Let $c : type \in OP$ be a constant symbol. Then we denote the term $c(\langle\rangle)$ as in classical algebras by $c \in T_{\Sigma,type}(X)$. Moreover, let $op : (type_1 \to type_2) \in OP$ be a constant symbol and $term \in T_{\Sigma,type_1}$. Then we use $(op.term)$ as an abbreviation of $apply^{type_1,type_2}(op, term)$, or more precisely of $apply^{type_1,type_2}(op(\langle\rangle), term)$.

The set $T_{\Sigma,unit}(X)$ is the set of atomic formulas. Especially let $p' : type \rightharpoonup unit$ be a predicate symbol and $p : (type \to unit) \in OP$ be a predicate (see Rem. 3.1.9), then for all $term \in T_{\Sigma,type}$ we have $p'(term), (p.term) \in T_{\Sigma,unit}(X)$.

In the subsequent sections we introduce variables and operation symbols, which occur in the net inscription of algebraic higher-order nets. For this reason we introduce the notion of the set variables and the set of operation symbols occurring in terms.

**Definition 3.1.17 (Variables and Operation Symbols of Terms)**
Let a higher-order signature $\Sigma = (S, OP, X)$ and $term \in T_\Sigma(X)$ be given. Then the set $Var(term)$ of variables occurring in $term$ and the set $Op(term)$ of operation symbols occurring in $term$ are recursively defined by

1. $Var(\langle\rangle) = Op(\langle\rangle) = \emptyset$ for $\langle\rangle \in T_{\Sigma,unit}(X)$
2. $Var(x) = \{x\}$ and $Op(x) = \emptyset$ for all $x : type \in X$
3. $Var(op(term)) = Var(term)$ and $Op(op(term)) = \{op\} \cup Op(term)$
   for all $(op : type_1 \rightharpoonup type_2) \in OP, term \in T_{\Sigma,type_1}(X)$
4. $Var(\langle term_1, \ldots, term_n\rangle) = Var(term_1) \cup \ldots \cup Var(term_n)$ and
   $Op(\langle term_1, \ldots, term_n\rangle) = Op(term_1) \cup \ldots \cup Op(term_n)$
   for all $term_1 \in T_{\Sigma,type_1}(X), \ldots, term_n \in T_{\Sigma,type_n}(X)$
5. $Var(pr_i^{(type_1\star\ldots\star type_n)}(term)) = Var(term)$ and
   $Op(pr_i^{(type_1\star\ldots\star type_n)}(term)) = Op(term)$
   for all $pr_i^{(type_1\star\ldots\star type_n)} \in (OP^{\to} \setminus OP), term \in T_{\Sigma,(type_1\star\ldots\star type_n)}(X)$.

Because algebraic higher-order nets are related by mappings, we need a notion of translation of terms to relate the net inscriptions of the source net to the net inscriptions of the target net. Note that in the translation we consider the sets of terms wrt. two different higher-order signatures.

**Definition 3.1.18 (Translation of Terms)**
Let $f_\Sigma : \Sigma_1 \longrightarrow \Sigma_2$ be a higher-order signature morphism with $f = (f_S, f_{OP}, f_X)$. The translation of $\Sigma_1$-terms by $f_\Sigma$ to $\Sigma_2$-terms is obtained by replacing the corresponding variables and operation symbols, i.e. $f_\Sigma^\sharp : T_{\Sigma_1}(X_1) \longrightarrow T_{\Sigma_2}(X_2)$ is recursively defined by

1. $f_\Sigma^\sharp(\langle\rangle) = \langle\rangle$ for $\langle\rangle \in T_{\Sigma_1,unit}(X)$
2. $f_\Sigma^\sharp(x) = f_X(x)$ for all $x : type \in X_1$
3. $f_\Sigma^\sharp(op(term)) = f_{OP}(op)(f_\Sigma^\sharp(term))$
   for all $(op : type_1 \rightharpoonup type_2) \in OP_1, term \in T_{\Sigma_1,type_1}(X_1)$
4. $f_\Sigma^\sharp(\langle term_1, \ldots, term_n\rangle) = \langle f_\Sigma^\sharp(term_1), \ldots, f_\Sigma^\sharp(term_n)\rangle$
   for all $term_1 \in T_{\Sigma_1,type_1}(X_1), \ldots, term_n \in T_{\Sigma_1,type_n}(X_1)$
5. $f_\Sigma^\sharp(pr_i^{(type_1 \star \ldots \star type_n)}(term)) = pr_i^{(f_S^{\rightarrow}(type_1 \star \ldots \star type_n))}(f_\Sigma^\sharp(term))$
   for all $pr_i^{(type_1 \star \ldots \star type_n)} \in (OP_1^{\rightarrow} \setminus OP_1), term \in T_{\Sigma_1,(type_1 \star \ldots \star type_n)}(X_1)$.

The set of terms wrt. a higher-order signature leads to a functorial construction. This result will be used to show the finitely cocompleteness of the category of algebraic higher-order net schemes in Section 4.1.

**Fact 3.1.19 (Functor $T_\Sigma : \mathbf{HOSig} \to \mathbf{Sets}$)**
The functor $T_\Sigma : \mathbf{HOSig} \to \mathbf{Sets}$ is defined for all higher-order signatures $\Sigma = (S, OP, X)$ by the corresponding set of terms $T_\Sigma(X)$ and for all $f_\Sigma : \Sigma_1 \longrightarrow \Sigma_2$ by the translation of terms.

**Proof:** We have to show that $T_\Sigma$ preserves compositions and identities.

**Composition:** For all $f_{\Sigma,1} \in Mor_{\mathbf{HOSig}}(\Sigma_1, \Sigma_2)$ and $f_{\Sigma,2} \in Mor_{\mathbf{HOSig}}(\Sigma_2, \Sigma_3)$ we have

$$\begin{aligned} T_\Sigma(f_{\Sigma,2} \circ f_{\Sigma,1}) &= (f_{\Sigma,2} \circ f_{\Sigma,1})^\sharp \\ &= (f_{\Sigma,2}^\sharp \circ f_{\Sigma,1}^\sharp) \\ &= T_\Sigma(f_{\Sigma,2}) \circ T_\Sigma(f_{\Sigma,1}) \end{aligned}$$

where $(f_{\Sigma,2} \circ f_{\Sigma,1})^\sharp = f_{\Sigma,2}^\sharp \circ f_{\Sigma,1}^\sharp$ can be shown by structural induction over the term structure $T_\Sigma(X)$.

**Induction Base:**

1. $(f_{\Sigma,2}^\sharp \circ f_{\Sigma,1}^\sharp)(\langle\rangle) = \langle\rangle = (f_{\Sigma,2} \circ f_{\Sigma,1})^\sharp(\langle\rangle)$
   for $\langle\rangle \in T_{\Sigma_1,unit}(X_1)$
2. $(f_{\Sigma,2}^\sharp \circ f_{\Sigma,1}^\sharp)(x) = f_{X,2}(f_{X,1}(x)) = (f_{\Sigma,2} \circ f_{\Sigma,1})^\sharp(x)$
   for all $x \in X_1$

**Induction Step:** Let $(f_{\Sigma,2} \circ f_{\Sigma,1})^\sharp(term_i) = (f_{\Sigma,2}^\sharp \circ f_{\Sigma,1}^\sharp)(term_i)$ for some $term_i \in T_\Sigma(X), i = 1, \ldots, n$.

3. $$\begin{aligned} &(f_{\Sigma,2}^\sharp \circ f_{\Sigma,1}^\sharp)(op(term)) \\ &= f_{OP,2}(f_{OP,1}(op))((f_{\Sigma,2}^\sharp \circ f_{\Sigma,1}^\sharp)(term)) \\ &= f_{OP,2}(f_{OP,1}(op))((f_{\Sigma,2} \circ f_{\Sigma,1})^\sharp(term)) \\ &= (f_{\Sigma,2} \circ f_{\Sigma,1})^\sharp(op(term)) \end{aligned}$$
   for all $op : type_1 \rightharpoonup type_2 \in OP$ and all $term \in T_{\Sigma,type_1}(X))$

**4.** $(f^\sharp_{\Sigma,2} \circ f^\sharp_{\Sigma,1})(\langle term_1, \ldots, term_n\rangle)$
$= \langle (f^\sharp_{\Sigma,2} \circ f^\sharp_{\Sigma,1})(term_1), \ldots, (f^\sharp_{\Sigma,2} \circ f^\sharp_{\Sigma,1})(term_n)\rangle$
$= \langle (f_{\Sigma,2} \circ f_{\Sigma,1})^\sharp(term_1), \ldots, (f_{\Sigma,2} \circ f_{\Sigma,1})^\sharp(term_n)\rangle$
$= (f_{\Sigma,2} \circ f_{\Sigma,1})^\sharp(\langle term_1, \ldots, term_n\rangle)$
for all $term_1 \in T_{\Sigma,type_1}(X), \ldots, term_n \in T_{\Sigma,type_n}(X)$

**5.** $(f^\sharp_{\Sigma,2} \circ f^\sharp_{\Sigma,1})(pr_i^{(type_1 \star \ldots \star type_n)}(term))$
$= pr_i^{((f^{\rightarrow}_{S,2} \circ f^{\rightarrow}_{S,1})(type_1) \star \ldots \star type_n))}((f^\sharp_{\Sigma,2} \circ f^\sharp_{\Sigma,1})(term))$
$= pr_i^{((f^{\rightarrow}_{S,2} \circ f^{\rightarrow}_{S,1})(type_1 \star \ldots \star type_n))}((f_{\Sigma,2} \circ f_{\Sigma,1})^\sharp(term))$
$= (f_{\Sigma,2} \circ f_{\Sigma,1})^\sharp(pr_i^{(type_1 \star \ldots \star type_n)}(term))$
for all $pr_i^{(type_1 \star \ldots \star type_n)} \in (OP^{\rightarrow} \setminus OP)$,
$term \in T_{\Sigma_1,(type_1 \star \ldots \star type_n)}$

**Identity:** For all $id_\Sigma \in Mor_{\mathbf{HOSig}}(\Sigma, \Sigma)$ we have

$$T_\Sigma(id_\Sigma) = id^\sharp_\Sigma = id_{T_\Sigma(X)}.$$

□

In the following we introduce higher-order partial algebras, which provide a set theoretic definition of domains and operations. Operation symbols named by the higher-order signature are interpreted by partial functions. As mentioned above we prefer an intensional setting, i.e. higher-order types and especially function types are interpreted by arbitrary sets.

**Definition 3.1.20 (Higher-Order Partial Algebras)**
Let $\Sigma = (S, OP, X)$ be a higher-order signature. A higher-order (partial) $\Sigma$-algebra $A = (A(BFS^{\rightarrow}), A(OP))$ is given by a $BFS^{\rightarrow}$-sorted set of carriers of $A$:

$$A(BFS^{\rightarrow}) = (A_{type})_{type \in BFS^{\rightarrow}}$$

and by an $OP$-sorted set of partial operations of $A$, i.e. there is a partial function $op_A : A_{type_1} \nrightarrow A_{type_2}$ for every $op : type_1 \rightharpoonup type_2 \in OP$:

$$A(OP) = (op_A)_{op \in OP}.$$

The carriers $A(BFS^{\rightarrow})$ of $A$ can be extended inductively to an $S^{\rightarrow}$-sorted set $A(S^{\rightarrow}) = (A_{type})_{type \in S^{\rightarrow}}$ as follows:

$$A_{unit} := \{()\} \text{ and}$$
$$A_{(type_1 \star \ldots \star type_n)} := A_{type_1} \times \ldots \times A_{type_n}.$$

Moreover, the set of partial operations $A(OP)$ of $A$ can be extended to an $OP^{\rightarrow}$-sorted set $A(OP^{\rightarrow}) = (op_A)_{op \in OP^{\rightarrow}}$ by

$$pr_{i,A}^{(type_1 \star \ldots \star type_n)} : A_{(type_1 \star \ldots \star type_n)} \longrightarrow A_{type_i} \text{ with}$$
$$pr_{i,A}^{(type_1 \star \ldots \star type_n)}(a_1, \ldots, a_n) = a_i$$

for all $pr_i^{(type_1 \star \ldots \star type_n)} \in (OP^{\rightarrow} \setminus OP)$ and $(a_1, \ldots, a_n) \in A_{(type_1 \star \ldots \star type_n)}$.

**Definition 3.1.21 (Higher-Order Term Evaluation)**
Let $\Sigma = (S, OP, X)$ be a higher-order signature. A variable valuation of $X$ in a higher-order partial $\Sigma$-algebra $A$ is a (total) function $v : X \longrightarrow A(S^{\rightarrow})$ with $v(x) \in A_{type}$ for $x : type \in X$. The term evaluation of $T_\Sigma(X)$ in $A(S^{\rightarrow})$ wrt. $v$ is a partial function $v^\sharp : T_\Sigma(X) \multimap A$ recursively defined by

**1.** $v^\sharp(\langle\rangle) \stackrel{S}{=} ()$ for $\langle\rangle \in T_{\Sigma,unit}(X)$

**2.** $v^\sharp(x) \stackrel{S}{=} v(x)$ for all $x \in X$

**3.** $v^\sharp(op(term)) \stackrel{S}{=} op_A(v^\sharp(term))$
for all $op : type_1 \rightharpoonup type_2 \in OP$ and $term \in T_{\Sigma,type_1}(X)$

**4.** $v^\sharp(\langle term_1, \ldots, term_n\rangle) \stackrel{S}{=} (v^\sharp(term_1), \ldots, v^\sharp(term_n))$
for all $term_1 \in T_{\Sigma,type_1}(X), \ldots, term_n \in T_{\Sigma,type_n}(X)$

**5.** $v^\sharp(pr_i^{(type_1 \star \ldots \star type_n)}(term)) \stackrel{S}{=} pr_{i,A}^{(type_1 \star \ldots \star type_n)}(v^\sharp(term))$
for all $pr_i^{(type_1 \star \ldots \star type_n)} \in (OP^{\rightarrow} \setminus OP)$ and $term \in T_{\Sigma_1,(type_1 \star \ldots \star type_n)}$

were the equality symbol $exp \stackrel{S}{=} exp'$ is used to state strong equality, i.e. that either both sides denote the same value or both sides do not denote any value.

**Remark 3.1.22 (Application Symbol)**
We use the notation $(op.term)^A$ as an abbreviation of $apply_A^{type_1,type_2}(op, term)$. Moreover, let $c : type \in OP$ be a constant symbol. Then $c_A = a \in A_{type}$ is an abbreviation of $c_A : A_{unit} \longrightarrow A_{type}$ with $c_A(()) = a$.

**Definition 3.1.23 (Higher-Order Homomorphisms)**
Let $\Sigma = (S, OP, X)$ be a higher-order signature. For two higher-order partial $\Sigma$-algebras $A_1$ and $A_2$ a higher-order homomorphism is given by a total $BFS^{\rightarrow}$-sorted function

$$f_A : A_1 \longrightarrow A_2 = (f_{A,type} : A_{1,type} \longrightarrow A_{2,type})_{type \in BFS^{\rightarrow}}$$

satisfying the following two conditions for all $op : type_1 \rightharpoonup type_2 \in OP$:

**(D)** $f_{A,type_1}(dom(op_{A_1})) \subseteq dom(op_{A_2})$

**(H)** $f_{A,type_2}(op_{A_1}(a)) = op_{A_2}(f_{A,type_1}(a))$ for all $a \in dom(op_{A_1}) \subseteq A_{1,type_1}$

Every $BFS^{\rightarrow}$-sorted function can be extended to a $S^{\rightarrow}$-sorted function

$$f_A : A_1 \longrightarrow A_2 = (f_{A,type} : A_{1,type} \longrightarrow A_{2,type})_{type \in S^{\rightarrow}}$$

by

$$f_{A,unit} := id_{\{()\}} \text{ and}$$
$$f_{A,(type_1 \star \ldots \star type_n)}(a_1, \ldots, a_n) := (f_{A,type_1}(a_1), \ldots, f_{A,type_n}(a_n))$$

for all $(a_1, \ldots, a_n) \in A_{1,(type_1 \star \ldots \star type_n)} \neq \emptyset$. In case $A_{1,type_i} = \emptyset$ for some $1 \leq i \leq n$ $f_{A,(type_1 \star \ldots \star type_n)}$ is considered to be the inclusion of the empty set $A_{1,(type_1 \star \ldots \star type_n)} = \emptyset$ into $A_{2,(type_1 \star \ldots \star type_n)}$.

**Fact 3.1.24 (Evaluation Theorem)**
For any variable valuation $v : X \longrightarrow A_1(S^{\rightarrow})$ and any higher-order homomorphism $f_A : A_1 \longrightarrow A_2$ we have that $f_A \circ v^\sharp$ is a restriction of $(f_A \circ v)^\sharp$, i.e.

**(1)** $dom(v^\sharp) = dom(f_A \circ v^\sharp) \subseteq dom((f_A \circ v)^\sharp)$ and

**(2)** $(f_A \circ v^\sharp)(term) = (f_A \circ v)^\sharp(term)$ for all $term \in dom(f_A \circ v^\sharp)$.

**Proof:** Let $v^\sharp : T_\Sigma \xrightarrow{\circ} A$ be a term evaluation. Then we have

$$dom(v^\sharp) = dom(f_A \circ v^\sharp)$$

because $f_A$ is a total $S^\rightarrow$-sorted function. Moreover, we have by induction over the term structure $T_\Sigma(X)$:

**Induction Base:**

**1.** $f_A \circ v^\sharp(\langle\rangle) = f_A(()) = (f_A \circ v)^\sharp(\langle\rangle)$
for $\langle\rangle \in T_{\Sigma,unit}(X)$ and $\langle\rangle \in dom(f_A \circ v^\sharp)$

**2.** $f_A \circ v^\sharp(x) = f_A \circ v(x) = (f_A \circ v)^\sharp(x)$
for all $x \in X$ and $x \in dom(f_A \circ v^\sharp)$

**Induction Step:** Given $f_A \circ v^\sharp(term_i) = (f_A \circ v)^\sharp(term_i)$ for $i \in \{1, \ldots, n\}$ and $term_i \in T_\Sigma(X)$ and $term_i \in dom(f_A \circ v^\sharp)$.

**3.** $$\begin{aligned} f_A \circ v^\sharp(op(term)) &= f_A(op_{A_1}(v^\sharp(term)) \\ &= op_{A_2}(f_A \circ v^\sharp(term)) \\ &= op_{A_2}((f_A \circ v)^\sharp(term)) \\ &= (f_A \circ v)^\sharp(op(term)) \end{aligned}$$
for all $op : type_1 \rightharpoonup type_2 \in OP$ and $term \in T_{\Sigma,type_1}(X))$
and $op(term) \in dom(f_A \circ v^\sharp)$

**4.** $$\begin{aligned} f_A \circ v^\sharp(\langle term_1, \ldots, term_n\rangle) &= f_A(v^\sharp(term_1), \ldots, v^\sharp(term_n)) \\ &= (f_A \circ v^\sharp(term_1), \ldots, f_A \circ v^\sharp(term_n)) \\ &= ((f_A \circ v)^\sharp(term_1), \ldots, (f_A \circ v)^\sharp(term_n)) \\ &= (f_A \circ v)^\sharp(\langle term_1, \ldots, term_n\rangle) \end{aligned}$$
for all $term_1 \in T_{\Sigma,type_1}(X), \ldots, term_n \in T_{\Sigma,type_n}(X)$
and $term_1 \in dom(f_A \circ v^\sharp), \ldots, term_n \in dom(f_A \circ v^\sharp)$

**5.** $$\begin{aligned} f_A \circ v^\sharp(pr_i^{(type_1 \star \ldots \star type_n)}(term)) &= pr_{i,A_2}^{(type_1 \star \ldots \star type_n)}(f_A \circ v^\sharp(term)) \\ &= pr_{i,A_2}^{(type_1 \star \ldots \star type_n)}((f_A \circ v)^\sharp(term)) \\ &= (f_A \circ v)^\sharp(pr_i^{(type_1 \star \ldots \star type_n)}(term)) \end{aligned}$$
for all $term \in T_{\Sigma_1,(type_1 \star \ldots \star type_n)}(X)$ with $term \in dom(f_A \circ v^\sharp)$.

Thus $dom(f_A \circ v^\sharp) \subseteq dom(f_A \circ v)^\sharp$ and $f_A \circ v^\sharp(term) = (f_A \circ v)^\sharp(term)$ for all $term \in dom(f_A \circ v^\sharp)$. □

**Fact 3.1.25 (Category HOAlg(Σ))**
The category **HOAlg(Σ)** consists of partial higher-order Σ-algebras as objects and higher-order homomorphisms as morphisms.

**Proof:** We show that the composition defined componentwise is associative and the identity law is satisfied.

**Composition:** Given higher-order partial $\Sigma$-algebras $A_i$ for $i \in \{1,2,3\}$ and a pair of higher-order homomorphisms

$$f_{A,1} =: A_1 \longrightarrow A_2 \text{ and } f_{A,2} : A_2 \longrightarrow A_3.$$

The composition of $f_{A,1}$ and $f_{A,2}$ defined componentwise by

$$\begin{array}{l} f_{A,2} \circ f_{A,1} : A_1 \longrightarrow A_3 \text{ with} \\ f_{A,2} \circ f_{A,1} = (f_{A,2,type} \circ f_{A,1,type} : A_{1,type} \longrightarrow A_{2,type})_{type \in S^{\rightarrow}} \end{array} .$$

is a higher-order homomorphism because functions are closed under composition and the conditions **(D)** and **(H)** in Def. 3.1.23 are satisfied.

$$\begin{array}{ll} f_{A,2,type_1}(f_{A,1,type_1}(dom(op_{A_1}))) & \subseteq f_{A,2,type_1}(dom(op_{A_2})) \\ & \subseteq dom(op_{A_3}) \end{array}$$

$$\begin{array}{ll} f_{A,2,type_2}(f_{A,1,type_2}(op_{A_1}(a))) & = f_{A,2,type_2}(op_{A_2}(f_{A,1,type_1}(a))) \\ & = op_{A_3}(f_{A,2,type_1}(f_{A,1,type_1}(a))) \end{array}$$

for all $op : type_1 \rightharpoonup type_2 \in OP$ and for all $a \in dom(op_{A_1}) \subseteq A_{1,type_1}$.

**Associativity:** Let $f_{A,i}$ for $i \in \{1,2,3\}$ be three higher-order homomorphisms. Then the composition is associative due to the associativity of the $S^{\rightarrow}$-sorted functions, i.e. we have

$$(f_{A,3} \circ f_{A,2}) \circ f_{A,1} = f_{A,3} \circ (f_{A,2} \circ f_{A,1}).$$

**Identity:** The identity is given by the identity higher-order homomorphisms and satisfies obviously the identity law, i.e. we have for $f_A : A \to A'$

$$f_A \circ id_A = f_A \text{ and } id_{A'} \circ f_A = f_A.$$

□

## 3.2 Definition of Algebraic Higher-Order Nets

Based on the data type part provided in the previous section we introduce in this section algebraic higher-order nets schemes and algebraic higher-order nets, where the latter are algebraic higher-order net schemes with a suitable higher-order algebra. Algebraic higher-order nets are staying close to algebraic high-level nets as presented in Section 2.3. The relation between algebraic high-level nets and algebraic higher-order nets will be discussed later in Section 10.2.

In the following we present the main items, which are essential for algebraic higher-order nets, in more detail. Summarizing, net inscriptions are defined by terms over a higher-order signatures, while firing conditions are given by atomic formulas to guarantee, that certain constraints are respected. The marking of algebraic higher-order nets consists not only of simple data tokens but also of higher-order data tokens, which are elements from a given higher-order partial algebra. Note that the typing of places includes function and product types and the order of these types is allowed to be arbitrarily high.

**Partiality** Because we have introduced the more powerful framework based on the (possibly) partial interpretation of operation symbols, the concept of partiality also occurs in our approach of algebraic higher-order nets. In detail, the evaluation of the net inscriptions may not yield a result. Thus, we only consider those variable valuations for which the evaluation of the net inscriptions is defined. From a practical point of view the advantage is that in our approach there is no need for an explicit error handling on the level of algebraic higher-order nets and a lot of firing conditions can be avoided. More precisely there is a shift from the model level to the data type level caused by the observation that both - firing conditions and partial operations - navigate the behavior of our models. Thus instead of using firing conditions in most cases we are able to specify suitable partial operations for the same purpose.

**Function Types** The concept of higher-order functions can be used to define the net inscriptions in algebraic higher-order nets. Because we prefer the notion of intensional function space, in our approach functions occur as tokens in a natural way and we can distinguish between different function tokens, which realize the same behavior. Due to function types the mechanism of operation late-binding mentioned above is introduced into our concept.

**Product Types** Our data type part introduces product types. This makes not only higher-order algebras more applicable for the user but also algebraic higher-order nets because we allow the assignment of products types to places. Moreover, we obtain a more compact description of models as we will see in Section 5.2.

**Predicates** In higher-order partial algebras predicates are fully determined by partial functions into a singleton set. Thus, we obtain the notion of atomic formulas, which is used in algebraic higher-order nets for the definition of firing conditions. Because we are able to define arbitrary predicates, the complexity of firing conditions can be arbitrarily high.

Next we define algebraic higher-order net schemes, where in contrast to algebraic high-level nets the data type part is given by higher-order signatures. For most applications it is sufficient to consider a specific signature and define a corresponding algebra. Moreover, it is more likely to get some results for algebraic higher-order nets with higher-order signatures instead of higher-order specifications. Nevertheless, possible extensions to algebraic higher-order specifications will be discussed in Chapter 7.

**Definition 3.2.1 (Algebraic Higher-Order Net Schemes)**
An algebraic higher-order (AHO) net scheme

$$N = (\Sigma, P, T, pre, post, cond, type)$$

consists of

- a higher-order signature with variables $\Sigma = (S, OP, X)$ (see Def. 3.1.10),
- a set $P$ of places and a set $T$ of transitions,
- pre- and post domain functions

  $$pre, post : T \longrightarrow (T_\Sigma(X) \otimes P)^\oplus$$

  assigning to each transition $t \in T$ the pre- and post domains $pre(t)$ and $post(t)$ (see Rem. 3.2.2), respectively,

- a firing condition function

$$cond : T \longrightarrow T_{\Sigma,unit}(X)$$

assigning to each transition $t \in T$ an atomic formula $cond(t)$ (see Rem. 3.1.16),

- and a type function

$$type : P \longrightarrow S^{\rightarrow}$$

assigning to each place $p \in P$ a higher-order type $type(p)$ (see Def. 3.1.4).

**Remark 3.2.2 (Pre- and Post-Domain Function)**
Denoting by $T_\Sigma(X)$ the set of terms over the signature $\Sigma$ (see Def. 3.1.14) and by $M^\oplus$ the free commutative monoid over a set $M$ (see Appendix B), the set of all type consistent arc inscriptions $(T_\Sigma(X) \otimes P)$ is defined by

$$(T_\Sigma(X) \otimes P) = \{(term, p) | term \in T_{\Sigma,type(p)}(X), p \in P\}.$$

Thus, $pre(t)$ (and similar $post(t)$) is of the form

$$pre(t) = \sum_{i=1}^{n} (term_i, p_i) \text{ for } n \in \mathbb{N}.$$

This means $p_i$ is in the pre domain of $t$ with arc-inscription $term_i$ for $i \in \{1, \ldots, n\}$. Note that $\sum_{i=1}^{n}(term_i, p_i)$ might not be in normal form, but can be transformed into normal form (see Appendix B). So, we distinguish the following four cases for all $t \in T$ and $p \in P$, where the notion $pre(t)_{|p}$ is the restriction of the pre domain of the transition $t$ to the place $p$. For further details we refer to Appendix B.

- $pre(t)_{|p} = \lambda$, i.e. there is no arc from $p$ to $t$ (empty case),
- $pre(t)_{|p} = term$, i.e. there is one and only one arc-inscription $term$ for the arc from $p$ to $t$ (unary case),
- $pre(t)_{|p} = \sum_{j=1}^{m} term_j, m \geq 2$, and $term_1 \neq \ldots \neq term_m$, i.e. there is an arc-inscription $term_1 \oplus \ldots \oplus term_m$ for the arc from $p$ to $t$ (multi-case I),
- $pre(t)_{|p} = m \cdot term, m \in \mathbb{N}, m \geq 2$, i.e. there is an arc-inscription $m \cdot term$ with coefficient $m$ for the arc from $p$ to $t$ (multi case II),

and analogously for the post domain. Note that multi-case I and multi-case II might be mixed, e.g. $pre(t)_{|p} = term_1 \oplus (3 \cdot term_2)$ for $term_1 \neq term_2$.

**Definition 3.2.3 (Pre- and Post Sets)**
Let $N = (\Sigma, P, T, pre, post, cond, type)$ be an AHO-net scheme and $t \in T$ with $pre(t) = \sum_{i=1}^{n}(term_i, p_i)$. Then the pre set of $t$ is defined by

$$\bullet t = \{p_1, \ldots, p_n\}$$

and analogously for the post set of $t$ denoted by $t\bullet$. The pre set for some $p \in P$ is defined by

$$\bullet p = \{t | t \in T, pre(t)_{|p} \neq \lambda\}$$

and analogously for the post set of $p$ denoted by $p\bullet$.

In the subsequent chapters we frequently require a specific net structure, called contextual place. Exactly one transition is adjacent to a contextual place, where the inscriptions for arcs between this transition and the contextual place are given by the same variable. Thus tokens on contextual places are recovered in each firing step. In this way the definition is in some sense similar to that of contextual nets with read only arcs defined in [MR95] for the case of classical Petri nets. Additionally we assume that a contextual place is not shared with other transitions.

**Definition 3.2.4 (Contextual Place)**
Given an AHO-net scheme $N = (\Sigma, P, T, pre, post, cond, type)$ with $\Sigma = (S, OP, X)$ and a transition $t \in T$. Then a place $p \in P$ is a contextual place of $t$ if there exists a variable $x \in X$ such that

$$pre(t)_{|p} = post(t)_{|p} = x$$

and for all $t' \in T \setminus \{t\}$ we have

$$p \notin {}^\bullet t' \wedge p \notin t'^\bullet$$

Grounding on the notions in Def. 3.1.17, we define the set of variables and the set of operation symbols, which occur in the net inscription of algebraic higher-order nets. This will be used on the one hand to define the operational behavior in the following definitions and on the other hand for some constructions, which will be presented in Chapter 5.

**Definition 3.2.5 (Variables of Transitions)**
Let $N = (\Sigma, P, T, pre, post, cond, type)$ be an AHO-net scheme. We define the set $Var(t)$ of variables of a transition $t \in T$ as the set of all variables occurring in the pre- and post domain resp. firing condition of $t$, i.e.

$$Var(t) = Var(pre(t)) \cup Var(post(t)) \cup Var(cond(t))$$

where for $pre(t) = \sum_{i=1}^{n}(term_i, p_i)$ we define $Var(pre(t)) = \bigcup_{i\in\{1,\ldots,n\}} Var(term_i)$ and analogously $Var(post(t))$.

**Definition 3.2.6 (Operation Symbols of Transitions)**
Let $N = (\Sigma, P, T, pre, post, cond, type)$ be an AHO-net scheme. We define the set $Op(t)$ of operation symbols of a transition $t \in T$ as the set of all operation symbols occurring in the pre- and post domain resp. firing condition of $t$, i.e.

$$Op(t) = Op(pre(t)) \cup Op(post(t)) \cup Op(cond(t))$$

where for $pre(t) = \sum_{i=1}^{n}(term_i, p_i)$ we define $Op(pre(t)) = \bigcup_{i\in\{1,\ldots,n\}} Op(term_i)$ and analogously $Op(post(t))$.

In the following we define for each higher-order signature $\Sigma$ the class of algebraic higher-order nets wrt. $\Sigma$, where an algebraic higher-order net is an algebraic higher-order net scheme together with a higher-order partial $\Sigma$-algebra.

**Definition 3.2.7 (Algebraic Higher-Order Nets)**
Let $\Sigma$ be a higher-order signature. An algebraic higher-order (AHO) net $(N, A)$ wrt. $\Sigma$ consists of

- an AHO-net scheme $N = (\Sigma, P, T, pre, post, cond, type)$ and
- a higher-order partial $\Sigma$-algebra $A$ (see Def. 3.1.20).

The marking of an AHO-net is denoted by tuples consisting of a data element and the place it resides on. Moreover, we give a definition of parameterized markings, where the marking of some places is fixed.

**Definition 3.2.8 (Marking of an Algebraic Higher-Order Net)**
Given an AHO-net $(N, A)$ with the set of places $P$. Then a marking of $(N, A)$ is given by

$$M \in CP^{\oplus}$$

where

$$CP = (A \otimes P) = \{(a, p) | a \in A_{type(p)}, p \in P\}.$$

Let $P^F \subseteq P$ and $M(P^F) \in CP^{\oplus}$ be an arbitrary but fixed marking of the set of places $P^F$. Then the restriction of $CP^{\oplus}$ wrt. the marking $M(P^F)$ is defined by

$$CP^{\oplus}_{M(P^F)} = \{M \in CP^{\oplus} | M_{|P^F} = M(P^F)\}.$$

We write $M_{P^F}$ to indicate that $M_{P^F} \in CP^{\oplus}_{M(P^F)}$. Note that $M_{P^F}$ is not equal to $M(P^F)$ or $M_{|P^F}$.

Next we define the set of consistent transition valuations, i.e. under which conditions a transition $t$ can fire. In detail, the evaluation of the net inscriptions in the environment of the transition $t$ has to be defined.

**Definition 3.2.9 (Consistent Transition Valuation)**
Given an AHO-net $(N, A)$ with a set of transitions $T$. Let $t \in T$ a transition with variables $Var(t)$ and $v : Var(t) \longrightarrow A$ a variable valuation with term evaluation $v^\sharp : T_\Sigma(Var(t)) \multimap A$ (see Def. 3.1.21). Then $(t, v)$ is a consistent transition valuation iff

$$cond(t) \in dom(v^\sharp) \text{ and}$$
$$\forall (term, p) \in pre(t) \oplus post(t) : term \in dom(v^\sharp).$$

The set $CT$ of consistent transition valuations is defined by

$$CT = \{(t, v) | t \in T, v : Var(t) \longrightarrow A, (t, v) \text{ consistent transition valuation}\}.$$

**Definition 3.2.10 (Enabledness of Transitions)**
Given an AHO-net $(N, A)$ with a set of transitions $T$. Let $t \in T$ a transition with variables $Var(t)$ and $v : Var(t) \longrightarrow A$ a variable valuation such that $(t, v) \in CT$. The transition $t$ is enabled in a marking $M \in CP^{\oplus}$ under $v$, denoted by $M[(t, v)\rangle$, iff

$$pre_A(t, v) \leq M$$

where $pre_A : CT \longrightarrow CP^{\oplus}$ is defined by $pre_A(t, v) = \hat{v}(pre(t))$ and

$$\hat{v} : (T_\Sigma(Var(t)) \otimes P)^{\oplus} \longrightarrow (A \otimes P)^{\oplus}$$

is the unique extension of $v^\sharp : T_\Sigma(Var(t)) \longrightarrow A$ to terms and places, that is $\hat{v} = v^\sharp \times id_P$.

The behavior is given by the firing of transitions, i.e. tokens are moved over the set of places.

**Definition 3.2.11 (Follower Marking)**
Given an AHO-net $(N, A)$ with a set of transitions $T$. Let $t \in T$ a transition, $(t, v) \in CT$ a consistent transition valuation and $M \in CP^{\oplus}$ a marking such that $t$ be enabled in $M$ under $v$. Then the follower marking $M' \in CP^{\oplus}$ of $M$ after firing of $t$ is defined by

$$M' = M \ominus pre_A(t, v) \oplus post_A(t, v)$$

where $post_A : CT \longrightarrow CP^{\oplus}$ is defined by $post_A(t, v) = \hat{v}(post(t))$. We say that $M'$ is directly reachable from $M$ by the firing step $(t, v)$ denoted by $M[(t, v)\rangle M'$. The set $[M\rangle$ of reachable markings of $M \in CP^{\oplus}$ is the least set of markings such that

- $M \in [M\rangle$ and
- $M_1 \in [M\rangle \wedge M_1[(t, v)\rangle M_2 \Longrightarrow M_2 \in [M\rangle$.

A finite firing sequence

$$M_1[(t_1, v_1)\rangle M_2[(t_2, v_2)\rangle M_3 \ldots [(t_n, v_n)\rangle M_{n+1}$$

consist of a finite number $n \in \mathbb{N}$ of firing steps including the empty sequence $M_1[*\rangle M_1$ for $n = 0$. The length of the occurrence sequence is given by the number of steps $n$. The marking $M_1$ is called start marking and the marking $M_{n+1}$ is called end marking.

## 3.3 Example: Computation as AHO-Net

The algebraic high-level net "Computation" (see Fig. 2.2 in Section 2.3) can be represented in a more compact way by an algebraic higher-order net. To allow places resp. tokens of function and product types we have to replace the classical signature NAT by an appropriate higher-order signature and the classical NAT-algebra $A$ by a higher-order algebra. The signature NAT is turned into the higher-order signature HO-NAT by introducing the same sort *Nat* as a basic type of HO-NAT. In addition to this basic type the higher-order signature HO-NAT has product and function types like $Nat \star Nat, Nat \to Nat$ and $Pred(Nat \star Nat)$, where $Pred(Nat \star Nat)$ is an abbreviation of $(Nat \star Nat \to unit)$ (see Rem. 3.1.9).

The operations of the classical signature NAT are turned into constants of appropriate higher-order types, i.e. *leq* and *geq* are predicates of type $Pred(Nat \star Nat)$ and *add* and *sub* are constant symobls of type $(Nat \star Nat \to Nat)$.

HO-NAT =

| | |
|---|---|
| sorts: | $Nat$ |
| opns: | $leq, geq : Pred(Nat \star Nat)$ |
| | $add, sub : (Nat \star Nat \to Nat)$ |

Note that the predicates *leq* and *geq* of type $Pred(Nat \star Nat)$ are in fact operation symbols of type $unit \rightharpoonup (Nat \star Nat \to unit)$ and the constant symbols *add* and *sub* of type $(Nat \star Nat \to Nat)$ are in fact operation symbols of type $unit \rightharpoonup (Nat \star Nat \to Nat)$. Moreover, the higher-order signature HO-NAT contains an application symbol $apply^{type_1, type_2} : (type_1 \to type_2) \star type_1 \rightharpoonup type_2$ for each function type $(type_1 \to type_2) \in S^{\to}$, for instance $apply^{Nat \star Nat, Nat} : (Nat \star Nat \to Nat) \star (Nat \star Nat) \rightharpoonup Nat$ for the function type $(Nat \star Nat \to Nat) \in S^{\to}$. Because we have defined *leq* and *geq* as predicates there is no need for a boolean type and boolean constants especially not in this example.

In the higher-order HO-NAT-algebra *HO-A* the carrier $HO\text{-}A_{Nat}$ consists of natural numbers. Here we do not need an explicit error element *undef* because the interpretation of operation symbols in a higher-order partial algebra is given by partial functions. We define the higher-order HO-NAT-algebra *HO-A* by

- $HO\text{-}A_{Nat} = \mathbb{N}$,
- $HO\text{-}A_{Pred(Nat \star Nat)} = \{<, >\}$ with $leq_{HO\text{-}A} = <$, $geq_{HO\text{-}A} = >$, and

$$apply_{HO\text{-}A}^{Nat \star Nat, unit} : \{<, >\} \times (HO\text{-}A_{Nat} \times HO\text{-}A_{Nat}) \multimap HO\text{-}A_{unit}$$

for $n_1, n_2 \in \mathbb{N}$ with

$$(< .(n_1, n_2))^A = \begin{cases} () & \text{if } n_1 < n_2 \\ undef & \text{else} \end{cases}$$

and

$$(>.(n_1, n_2))^A = \begin{cases} () & \text{if } n_1 > n_2 \\ \mathit{undef} & \text{else} \end{cases}$$

- $\mathit{HO\text{-}A}_{Nat \star Nat \to Nat} = \{+, -\}$ with $\mathit{add}_{\mathit{HO\text{-}A}} = +$, $\mathit{sub}_{\mathit{HO\text{-}A}} = -$, and

$$\mathit{apply}_{\mathit{HO\text{-}A}}^{Nat \star Nat, Nat} : \{+, -\} \times (\mathit{HO\text{-}A}_{Nat} \times \mathit{HO\text{-}A}_{Nat}) \multimap \mathit{HO\text{-}A}_{Nat}$$

for $n_1, n_2 \in \mathbb{N}$ with

$$(+.(n_1, n_2))^A = n_1 + n_2$$

and

$$(-.(n_1, n_2))^A = \begin{cases} n_1 - n_2 & \text{if } n_1 > n_2 \\ \mathit{undef} & \text{else} \end{cases}$$

Note that the carrier of *HO-A* is inductively extended by $\mathit{HO\text{-}A}_{unit} := \{()\}$ and $\mathit{HO\text{-}A}_{type_1 \star \ldots \star type_n} := \mathit{HO\text{-}A}_{type_1} \times \ldots \times \mathit{HO\text{-}A}_{type_n}$. Moreover, for each type $type \in S^{\to}$, which is not interpreted above, we have $\mathit{HO\text{-}A}_{type} = \emptyset$ and for each application symbol $\mathit{apply}^{type_1, type_2} : (type_1 \to type_2) \star type_1 \rightharpoonup type_2$, which is not defined above, we have $(f.x)^A = \mathit{undef}$ for $f \in \mathit{HO\text{-}A}_{type_1 \to type_2}$ and $x \in \mathit{HO\text{-}A}_{type_1}$. Otherwise this results in an empty function.

The AHL-net "Computation" in Fig. 2.2 is turned into the AHO-net "Computation I" in Fig. 3.1, where the terms denoted in the net inscription of the AHL-net "Computation" are replaced by higher-order terms, e.g. the net inscription $\mathit{add}(x, y)$ is given by the higher-order term $(\mathit{add}.\langle x, y\rangle)$, which is an abbreviation of $\mathit{apply}^{Nat \star Nat, Nat}(\mathit{add}(\langle\rangle), \langle x, y\rangle)$. Moreover, the firing conditions of the AHL-net "Computation" in Fig. 2.2 are translated into atomic formulas. For instance the firing condition $\mathit{leq}(x, y) = tt$ is given by the atomic formula $(\mathit{leq}.\langle x, y\rangle)$. Thus the AHO-net "Computation I" is defined by $(N, \mathit{HO\text{-}A})$ with

$$N = ((\text{HO-NAT}, X), P, T, \mathit{pre}, \mathit{post}, \mathit{cond}, \mathit{type})$$

where $X$ is a suitable set of typed variable and

- $P = \{p_1, p_2, p_3\}$ is the set of places and $T = \{\mathit{compute\ add}, \mathit{compute\ sub}\}$ is the set of transitions,
- $\mathit{pre}, \mathit{post} : T \longrightarrow (T_{\text{HO-NAT}}(X) \otimes P)^{\oplus}$ are the pre- and post domain functions with

$$\mathit{pre}(t) = (x, p_1) \oplus (y, p_2)$$

and

$$\mathit{post}(t) = \begin{cases} (x, p_1) \oplus (y, p_2) \oplus ((\mathit{add}.\langle x, y\rangle), p_3) & \text{if } t = \mathit{compute\ add} \\ (x, p_1) \oplus (y, p_2) \oplus ((\mathit{sub}.\langle x, y\rangle), p_3) & \text{if } t = \mathit{compute\ sub} \end{cases}$$

- $\mathit{cond} : T \longrightarrow T_{\text{HO-NAT},unit}(X)$ is the firing condition function with

$$\mathit{cont}(t) = \begin{cases} (\mathit{leq}.\langle x, y\rangle) & \text{if } t = \mathit{compute\ add} \\ (\mathit{geq}.\langle x, y\rangle) & \text{if } t = \mathit{compute\ sub} \end{cases}$$

- and $\mathit{type} : P \longrightarrow S^{\to}$ is the type function with $\mathit{type}(p) = Nat$.

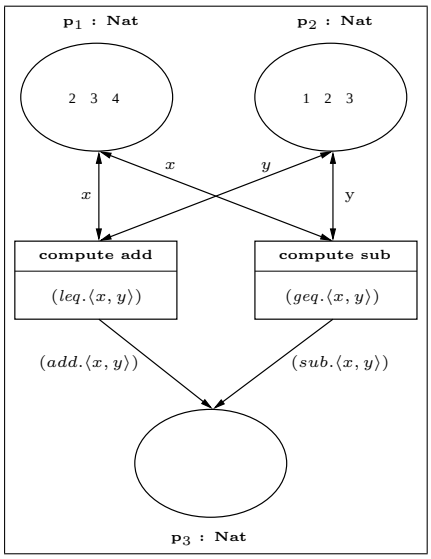



Figure 3.1: Algebraic higher-order net "Computation I"

As before the initial marking is given by $M = \sum_{i=2}^{4}(i, p_1) \oplus \sum_{j=1}^{3}(j, p_2)$. To demonstrate that the firing behavior of the AHL-net "Computation" and the AHO-net "Computation I" are equivalent we define a variable valuation

$$v : Var(\textit{compute sub}) \longrightarrow A \text{ with } v(x) = 3 \text{ and } v(y) = 2$$

for the variables of the transition *compute sub* (and similar for *compute add*). Next we check that $(\textit{compute sub}, v)$ is a consistent transition valuation, i.e. the net inscriptions of the transition *compute sub* are defined in *HO-A* under $v$.

- For $cond(t) = (geq.\langle x, y\rangle)$ we have

$$v^\sharp(geq.\langle x, y\rangle) = (> .(3, 2))^A = () \in \textit{HO-A}_{unit}.$$

- For $pre(t) = (x, p_1) \oplus (y, p_2)$ we have

$$v(x) = 3 \in \textit{HO-A}_{Nat} \text{ and } v(y) = 2 \in \textit{HO-A}_{Nat}.$$

- For $post(t) = ((sub.\langle x, y\rangle), p_3)$ we have

$$v^\sharp(sub.\langle x, y\rangle) = (-.(3, 2))^A = 1 \in \textit{HO-A}_{Nat}.$$

Moreover, the transition *compute sub* is enabled in $M$ under $v$ because

$$pre_A(t, v) = \widehat{v}((x, p_1) \oplus (y, p_2)) = (3, p_1) \oplus (2, p_2) \leq M.$$

Then the follower marking is computed as follows: the data element 3 on place $p_1$ and the data element 2 on place $p_2$ remain unchanged as indicated by the double arrow, while the result $v^\sharp(sub.\langle x, y\rangle) = 1$ is added to the place $p_3$.

Because in the higher-order signature HO-NAT there are product types like $Nat \star Nat$ available we get a more compact description of the AHO-net "Computation I" in Fig. 3.1 by applying a folding construction wrt. product types. This results in the AHO-net "Computation II" in Fig. 3.2. More precisely the folding construction wrt.

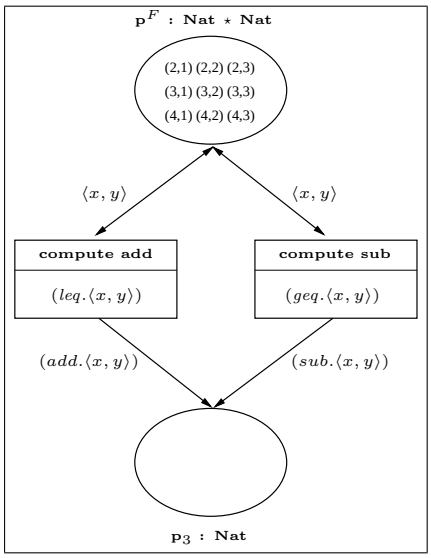



Figure 3.2: Algebraic higher-order net "Computation II"

product types is applied to the subset of places $P^F = \{p_1, p_2\} \subseteq P$. The set $P^F$ is called a set of uniform places because it is a finite set of at least two places having the same transitions in their pre- and post domains with unary arc inscriptions. Then the AHO-net "Computation I" in Fig. 3.1 is folded in the following way: the set of uniform places $P^F$ is replaced by the new place $p^F$ with type $Nat \star Nat$; the arc inscription in the environment of the new place $p^F$ are obtained by folding the arc inscriptions $x$ and $y$ into the tuple $\langle x, y\rangle$.

Note that the marking is regular, i.e. there is a subset $A_1 = \{2, 3, 4\} \subseteq \textit{HO-A}_{Nat}$ and a subset $A_2 = \{1, 2, 3\} \subseteq \textit{HO-A}_{Nat}$ so that $M_{|p_1} = \sum_{a_1 \in A_1} a_1$ and $M_{|p_2} = \sum_{a_2 \in A_2} a_2$. Thus we get the new marking of the place $p^F$ (see Fig. 3.2) by using the well-known Cartesian product, i.e. $M_{|p^F} = \sum_{(a_1,a_2) \in A_1 \times A_2} (a_1, a_2)$.

A main result of the folding construction is that both AHO-net "Computation I" and AHO-net "Computation II" are equivalent wrt. their firing behavior. There is a dual notion of an unfolding construction wrt. product types, i.e. the AHO-net "Computation II" can be unfolded into the AHO-net "Computation I".

Due to the notation of operation symbols like *add* and *sub* as constant symbols we are able to apply a folding construction wrt. constant symbols to the AHO-net "Computation II" in Fig. 3.2. Because operation symbols are first-class citizens like $+, - \in \textit{HO-A}_{(Nat \star Nat \to Nat)}$, they can be considered as tokens in an AHO-net. Moreover, we have variables of function type to build up terms like $(f_{add}.\langle x, y\rangle)$, where $f_{add}$ is a variable of type $(Nat \star Nat \to Nat)$ and we can define a variable valuation $v$ with $v(f_{add}) = +$.

The folding construction wrt. the constant symbol *add* and the transition *compute add* results in the AHO-net "Computation III" in Fig. 3.3. In detail we add a new place $p^{add}$ in the environment of the transition *compute add*, which is contextual in the sense that this place is not in the environment of other transitions and the arc inscriptions are given by the variable $f_{add}$. The type of the new place $p^{add}$ is given by the function type according to the constant symbol *add*, i.e. $type(p^{add}) = (Nat \star Nat \to Nat)$. Moreover, the net inscription in the post domain of the transition *compute add* is folded into the term $(f_{add}.\langle x, y\rangle)$, i.e. the constant symbol *add* is replaced by the variable $f_{add}$. If we use the constant $+$ as

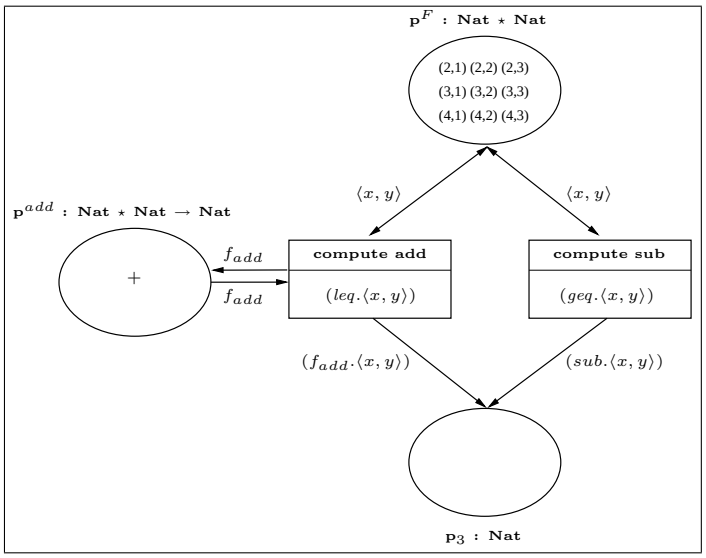



Figure 3.3: Algebraic higher-order net "Computation III"

initial marking for the new place $p^{add}$, the AHO-net "Computation III" and the AHO-net "Computation II" are equivalent wrt. their firing behavior.

In a further step we apply the folding construction wrt. the constant symbol *leq* and the transition *compute add*. Thus a new contextual place $p^{leq}$ is introduced with type $Pred(Nat \star Nat)$ and the firing condition of the transition *compute add* is folded into the term $(g_{leq}.\langle x, y\rangle)$. Afterwards we apply the folding construction wrt. product types and the set of uniform places $P^{F'} = \{p^{add}, p^{leq}\}$ and obtain the AHO-net "Computation IV" in Fig. 3.4.

Analogously to the folding constructions described above we can proceed the folding constructions wrt. the transition *compute sub* resulting in the AHO-net "Computation V" in Fig. 3.5. Note that all AHO-nets presented in this section up to now are still equivalent wrt. their firing behavior.

Finally, we use a horizontal structuring, called fusion, to obtain a more compact description of the AHO-net "Computation V" in Fig. 3.5. Intuitively spoken, a fusion of an AHO-net $N$ wrt. two copies of a subnet $N'$ in $N$ means that in the resulting AHO-net these copies are fused. The environment of the transitions *compute add* and *compute sub* are more or less identical in the AHO-net "Computation V" in Fig. 3.5. Thus in the AHO-net "Computation V" are two copies of a subnet consisting of a transition with a contextual place, the place $p^F$ and the place $p_3$ in its environment. Then we fuse the two transitions *compute add* and *compute sub* into one transition *compute* and the two contextual places $p^{F'}$ and $p^{F''}$ into one contextual place $p$, which results in the AHO-net "Computation VI" depicted in Fig. 3.6.

Note that in contrast to the folding construction given above the fusion is defined for AHO-net schemes. Thus we do not automatically obtain the initial marking as indicated in Fig. 3.6. Nevertheless, using this initial marking we can show that the AHO-net "Computation V" and the AHO-net "Computation VI" are equivalent wrt. their firing behavior.

Summarizing, AHO-nets give rise to a more compact and abstract representation of models due to the presence of function types and product types. The advantage of AHO-nets is the reusability of a fixed net structure. To include further computations

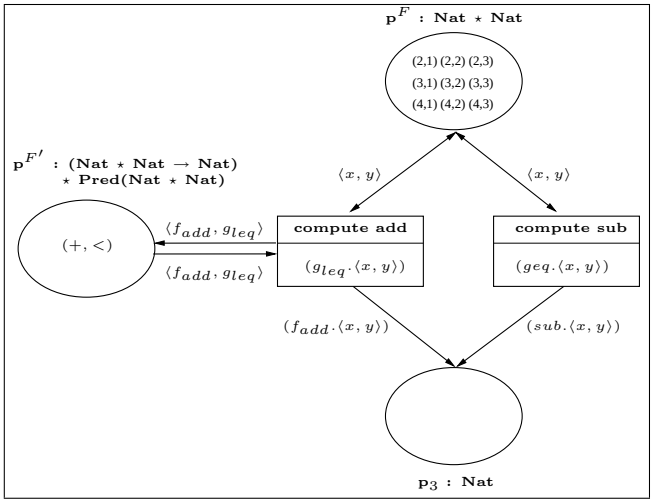



Figure 3.4: Algebraic higher-order net "Computation IV"

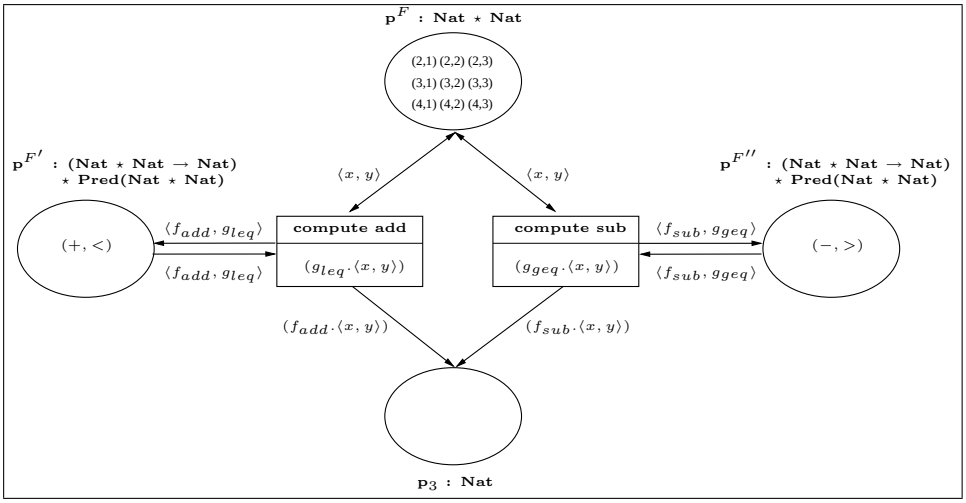



Figure 3.5: Algebraic higher-order net "Computation V"

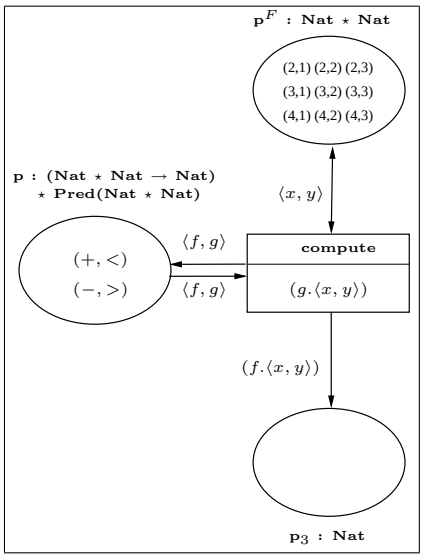



Figure 3.6: Algebraic higher-order net "Computation VI"

into the AHO-net "Computation VI" we change the variable part of the AHO-net, i.e. we change the marking of place $p$ but keep the same net structure. By contrast, in the AHL-net "Computation" in Fig. 2.2 the net structure have to be extended by one transition for each further computation. Thus AHO-nets lead to flexible and adaptable models and changes of the environment are reflected by changing the corresponding marking.

For a detailed definition of the folding and unfolding constructions we refer to Chapter 5, while the horizontal structuring technique fusion will be explained in the subsequent chapter. More examples can be found in our case study "Logistics" in Chapter 9.

# Chapter 4

# Structuring of Algebraic Higher-Order Nets

In this chapter we explore the formalisms of algebraic higher-order net schemes and algebraic higher-order nets on a categorical level. In general, category theory presents an abstract framework to model mathematical structures due to their universal properties so that they can be investigated independent from the internal structures. The basic categorical notions and results used in this chapter are summarized in Appendix A. In Section 3.1 we have also applied the framework of category theory to the concepts of higher-order algebras, where we mainly deal with the terms of categories, (adjoint) functors and colimits. By this we have achieved several important results with the most diverse influences on the theory in this chapter. To obtain the category of algebraic higher-order net schemes resp. algebraic higher-order nets we define structure preserving mappings, i.e. we give a suitable notion of net morphisms for these net classes.

In Section 4.1 we state our first main results of this chapter, which concerns horizontal structuring techniques of algebraic higher-order net schemes called union and fusion. In order to obtain these structuring techniques the existence of specific colimits like pushouts and coequalizers is essential. Union refers to the formal composition of model segments, while fusion is the identification of distinguished elements within a model. From a practical point of view the former technique supports the natural distribution and concurrency of model segments, while the latter leads to a more compact description of models. The main result presented in Section 4.2 shows the compatibility of the operational behavior with AHO-net morphisms, which is most important for the concept of a higher-order process semantics in Section 4.3. The higher-order process semantics is based on the notions of high-level net processes presented in [EHP+02]. The essential idea is to generalize the concept of occurrence nets from low-level to the higher-order case. Following this line of research higher-order processes have similar properties like low-level processes, but capture a set of different computations corresponding to different input parameters.

## 4.1 Horizontal Structuring of AHO-Net Schemes

Morphisms of AHO-net schemes are obtained by the combination of higher-order signature morphisms, mappings between the sets of places, and mappings between the sets of transitions. To achieve further results, these morphisms preserve the structure, i.e. the particular components of algebraic higher-order net schemes. In detail, higher-order signature morphisms are extended to translations of higher-

order terms, which have to preserve not only the net inscriptions in the pre- and post domains, but also the atomic formulas representing the firing conditions. Furthermore, the higher-order types assigned to the places have to be respected.

**Definition 4.1.1 (Algebraic Higher-Order Net Scheme Morphisms)**
Let two AHO-net schemes $N_i = (\Sigma_i, P_i, T_i, pre_i, post_i, cond_i, type_i)$ be given with higher-order signatures $\Sigma_i = (S_i, OP_i, X_i)$ for $i \in \{1, 2\}$. Then a AHO-net scheme morphism $f_N : N_1 \longrightarrow N_2$ is given by

$$f_N = (f_\Sigma, f_P, f_T)$$

with

- a higher-order signature morphism $f_\Sigma : \Sigma_1 \longrightarrow \Sigma_2$ (see Def. 3.1.11) with $f_\Sigma = (f_S : S_1 \longrightarrow S_2, f_{OP} : OP_1 \to OP_2, f_X : X_1 \to X_2)$,
- a function $f_P : P_1 \longrightarrow P_2$, and
- a function $f_t : T_1 \longrightarrow T_2$

such that the following diagrams commute componentwise:

$$\begin{array}{ccccc} T_{\Sigma_1,unit}(X_1) & \xleftarrow{cond_1} & T_1 & \underset{post_1}{\overset{pre_1}{\rightrightarrows}} & (T_{\Sigma_1}(X_1) \otimes P_1)^\oplus \\ \Big\downarrow{\scriptstyle f_\Sigma^\sharp} & = & \Big\downarrow{\scriptstyle f_T} & = & \Big\downarrow{\scriptstyle (f_\Sigma^\sharp \otimes f_P)^\oplus} \\ T_{\Sigma_2,unit}(X_2) & \xleftarrow[cond_2]{} & T_2 & \underset{post_2}{\overset{pre_2}{\rightrightarrows}} & (T_{\Sigma_2}(X_2) \otimes P_2)^\oplus \end{array}$$

$$\begin{array}{ccc} P_1 & \xrightarrow{type_1} & S_1^{\rightarrow} \\ \Big\downarrow{\scriptstyle f_P} & = & \Big\downarrow{\scriptstyle f_S^{\rightarrow}} \\ P_2 & \xrightarrow[type_2]{} & S_2^{\rightarrow} \end{array}$$

where $f_S^{\rightarrow} : S_1^{\rightarrow} \longrightarrow S_2^{\rightarrow}$ is the unique extension of $f_s : S_1 \longrightarrow S_2^{\rightarrow}$ (see Fact 3.1.7) and $f_s$ is induced by $f_S : S_1 \longrightarrow S_2$, i.e. $f_s(type) = f_S(type)$ for all basic types $type \in S_1$. Moreover, $f_\Sigma^\sharp : T_{\Sigma_1}(X_1) \longrightarrow T_{\Sigma_2}(X_2)$ is the translation of $\Sigma_1$-terms by $f_\Sigma$ to $\Sigma_2$-terms (see Def. 3.1.18) and $(f_\Sigma^\sharp \otimes f_P)$ denotes the corresponding function of type consisting arc inscriptions defined for all $(term, p) \in (T_{\Sigma_1}(X_1) \otimes P_1)$ by

$$(f_\Sigma^\sharp \otimes f_P)(term, p) = (f_\Sigma^\sharp(term), f_P(p))$$

and $(f_\Sigma^\sharp \otimes f_P)^\oplus$ is the unique homomorphic extension of $(f_\Sigma^\sharp \otimes f_P)$ (see Def. B.1.2 in the Appendix).

AHO-net schemes together with AHO-net scheme morphisms yield the category **AHOS**.

**Fact 4.1.2 (Category AHOS)**
The category **AHOS** consists of AHO-net schemes as objects and AHO-net scheme morphisms as morphisms.

**Proof:** We show that the composition defined componentwise is associative and the identity law is satisfied.

**Composition:** Given AHO-net schemes $N_i = (\Sigma_i, P_i, T_i, pre_i, post_i, cond_i, type_i)$ with $\Sigma_i = (S_i, OP_i, X_i)$ for $i \in \{1,2,3\}$ and a pair of AHO-net scheme morphisms

$$f_{N,1} = (f_{\Sigma,1}, f_{P,1}, f_{T,1}) : N_1 \longrightarrow N_2 \text{ and}$$
$$f_{N,2} = (f_{\Sigma,2}, f_{P,2}, f_{T,2}) : N_2 \longrightarrow N_3.$$

The composition of $f_{N,1}$ and $f_{N,2}$ defined componentwise by

$$f_{N,2} \circ f_{N,1} = (f_{\Sigma,2} \circ f_{\Sigma,1}, f_{P,2} \circ f_{P,1}, f_{T,2} \circ f_{T,1}) : N_1 \longrightarrow N_3$$

is an AHO-net scheme morphism, because signature morphisms are closed under composition (see Fact 3.1.12) and functions of places resp. transitions are closed under composition and

- the pre- and post domain functions are composable:

$$\begin{aligned}(f^\sharp_{\Sigma_2} \otimes f_{P_2})^\oplus \circ (f^\sharp_{\Sigma_1} \otimes f_{P_1})^\oplus \circ pre_1 &= (f^\sharp_{\Sigma_2} \otimes f_{P_2})^\oplus \circ pre_2 \circ f_{T_1} \\ &= pre_3 \circ f_{T_2} \circ f_{T_1}\end{aligned}$$

  (analogously for the post domain)

- the firing condition functions are composable:

$$\begin{aligned}f^\sharp_{\Sigma_2} \circ f^\sharp_{\Sigma_1} \circ cond_1 &= f^\sharp_{\Sigma_2} \circ cond_2 \circ f_{T_1} \\ &= cond_3 \circ f_{T_2} \circ f_{T_1}\end{aligned}$$

- and the type functions are composable:

$$\begin{aligned}f^\sharp_{\Sigma_2} \circ f^\sharp_{\Sigma_1} \circ type_1 &= f^\sharp_{\Sigma_2} \circ type_2 \circ f_{P_1} \\ &= type_3 \circ f_{P_2} \circ f_{P_1}.\end{aligned}$$

**Associativity** Let $f_{N,i} = (f_{\Sigma,i}, f_{P,i}, f_{T,i})$ for $i \in \{1,2,3\}$ be three AHO-net scheme morphisms. Then the composition is associative because it is defined componentwise, i.e. we have

$$(f_{N,3} \circ f_{N,2}) \circ f_{N,1} = f_{N,3} \circ (f_{N,2} \circ f_{N,1}).$$

**Identity** The identity is given by the identity AHO-net scheme morphism. Because the composition is defined componentwise, the identity law is satisfied, i.e. we have for $f_N = (f_\Sigma, f_P, f_T) : N \longrightarrow N'$

$$f_N \circ id_N = f_N \text{ and } id_{N'} \circ f_N = f_N.$$

□

We will now introduce two specific notions for horizontal structuring techniques, called union and fusion. The former allows the construction of larger models from model segments with shared subparts. Formally, the union is defined by the concept of pushouts. The application of the latter folds distinguished model segments into one model segment and is formally defined by the concept of coequalizers. The universal properties of these constructions imply that we obtain models, which are unique up to renaming. We mainly use the facts, that the category of higher-order signatures is (finitely) cocomplete (see Fact 3.1.13) and the category of sets is also (finitely) cocomplete. In the category of sets, for instance, pushouts are constructed by the disjoint union of two sets, but identifying the shared subsets.

**Theorem 4.1.3 (Union of AHO-nets)**
The union of two AHO-net schemes $N_1$ and $N_2$ via some interface AHO-net scheme $I$ with a pair of AHO-net scheme morphisms $i_{N,1} : I \longrightarrow N_1$ and $i_{N,2} : I \longrightarrow N_2$ is given by the pushout consisting of the pushout object $N$ and the pair of morphisms $f_N : N_1 \longrightarrow N$ and $g_N : N_2 \longrightarrow N$.

**Proof:** We have to show that **AHOS** has all pushouts. Given AHO-net schemes

$$N_i = (\Sigma_i, P_i, T_i, pre_i, post_i, cond_i, type_i) \text{ with } \Sigma_i = (S_i, OP_i, X_i)$$

for $i \in \{1, 2, 3\}$ and a pair of AHO-net scheme morphisms

$$f_{N,1} = (f_{\Sigma,1}, f_{P,1}, f_{T,1}) : N_1 \longrightarrow N_2 \text{ with}$$
$$f_{\Sigma,1} = (f_{S,1} : S_1 \longrightarrow S_2, f_{OP,1} : OP_1 \longrightarrow OP_2, f_{X,1} : X_1 \longrightarrow X_2)$$

and

$$g_{N,1} = (g_{\Sigma,1}, g_{P,1}, g_{T,1}) : N_1 \longrightarrow N_3 \text{ with}$$
$$g_{\Sigma,1} = (g_{S,1} : S_1 \longrightarrow S_3, g_{OP,1} : OP_1 \longrightarrow OP_3, g_{X,1} : X_1 \longrightarrow X_3).$$

Then the pushout consisting of the pushout object

$$N_4 = (\Sigma_4, P_4, T_4, pre_4, post_4, cond_4, type_4) \text{ with } \Sigma_4 = (S_4, OP_4, X_4)$$

and the pair of AHO-net scheme morphisms

$$f_{N,2} = (f_{\Sigma,2}, f_{P,2}, f_{T,2}) : N_2 \longrightarrow N_4 \text{ with}$$
$$f_{\Sigma,2} = (f_{S,2} : S_2 \longrightarrow S_4, f_{OP,2} : OP_2 \longrightarrow OP_4, f_{X,2} : X_2 \longrightarrow X_4)$$

and

$$g_{N,2} = (g_{\Sigma,2}, g_{P,2}, g_{T,2}) : N_3 \longrightarrow N_4 \text{ with}$$
$$g_{\Sigma,2} = (g_{S,2} : S_3 \longrightarrow S_4, g_{OP,2} : OP_3 \longrightarrow OP_4, g_{X,2} : X_3 \longrightarrow X_4)$$

is constructed by the following pushouts of higher-order signatures in **HOSig** (see Fact 3.1.13) and by the following pushouts of sets of places resp. transitions in **Sets**.

$$\begin{array}{ccc} \Sigma_1 & \xrightarrow{f_{\Sigma,1}} & \Sigma_2 \\ {\scriptstyle g_{\Sigma,1}}\downarrow & = & \downarrow{\scriptstyle f_{\Sigma,2}} \\ \Sigma_3 & \xrightarrow[g_{\Sigma,2}]{} & \Sigma_4 \end{array} \qquad \begin{array}{ccc} P_1 & \xrightarrow{f_{P,1}} & P_2 \\ {\scriptstyle g_{P,1}}\downarrow & = & \downarrow{\scriptstyle f_{P,2}} \\ P_3 & \xrightarrow[g_{P,2}]{} & P_4 \end{array} \qquad \begin{array}{ccc} T_1 & \xrightarrow{f_{T,1}} & T_2 \\ {\scriptstyle g_{T,1}}\downarrow & = & \downarrow{\scriptstyle f_{T,2}} \\ T_3 & \xrightarrow[g_{T,2}]{} & T_4 \end{array}$$

Moreover, we have due to Fact 3.1.7 the following pushout of higher-order types in **Sets**.

$$\begin{array}{ccc} S_1^{\rightarrow} & \xrightarrow{f^{\rightarrow}_{1,S}} & S_2^{\rightarrow} \\ {\scriptstyle g^{\rightarrow}_{1,S}}\downarrow & = & \downarrow{\scriptstyle f^{\rightarrow}_{2,S}} \\ S_3^{\rightarrow} & \xrightarrow[g^{\rightarrow}_{2,S}]{} & S_4^{\rightarrow} \end{array}$$

and due to Fact 3.1.19 the following commutative squares of higher-order terms in **Sets**.

$$\begin{array}{ccc} T_{\Sigma_1}(X_1) & \xrightarrow{f^\sharp_{\Sigma,1}} & T_{\Sigma_2}(X_2) \\ {\scriptstyle g^\sharp_{\Sigma,1}}\downarrow & = & \downarrow{\scriptstyle f^\sharp_{\Sigma,2}} \\ T_{\Sigma_3}(X_3) & \xrightarrow[g^\sharp_{\Sigma,2}]{} & T_{\Sigma_4}(X_4) \end{array}$$

$$\begin{array}{ccc} T_{\Sigma_1,unit}(X_1) & \xrightarrow{f^\sharp_{\Sigma,1}} & T_{\Sigma_2,unit}(X_2) \\ {\scriptstyle g^\sharp_{\Sigma,1}}\downarrow & = & \downarrow{\scriptstyle f^\sharp_{\Sigma,2}} \\ T_{\Sigma_3,unit}(X_3) & \xrightarrow[g^\sharp_{\Sigma,2}]{} & T_{\Sigma_4,unit}(X_4) \end{array}$$

Thus we have the following commutative square in **CMon** (see Def. B.1.4 in the Appendix).

$$\begin{array}{ccc} (T_{\Sigma_1}(X_1)\otimes P_1)^\oplus & \xrightarrow{(f^\sharp_{\Sigma,1}\otimes f_{P,1})^\oplus} & (T_{\Sigma_2}(X_2)\otimes P_2)^\oplus \\ {\scriptstyle (g^\sharp_{\Sigma,1}\otimes g_{P,1})^\oplus}\downarrow & = & \downarrow{\scriptstyle (f^\sharp_{\Sigma,2}\otimes f_{P,2})^\oplus} \\ (T_{\Sigma_3}(X_3)\otimes P_3)^\oplus & \xrightarrow[(g^\sharp_{\Sigma,2}\otimes g_{P,2})^\oplus]{} & (T_{\Sigma_4}(X_4)\otimes P_4)^\oplus \end{array}$$

Because $f_{N,1}$ and $g_{N,1}$ are AHO-net scheme morphisms we have

$$(f^\sharp_{\Sigma,2}\otimes f_{P,2})^\oplus \circ pre_2 \circ f_{T,1} = (g^\sharp_{\Sigma,2}\otimes g_{P,2})^\oplus \circ pre_3 \circ g_{T,1}.$$

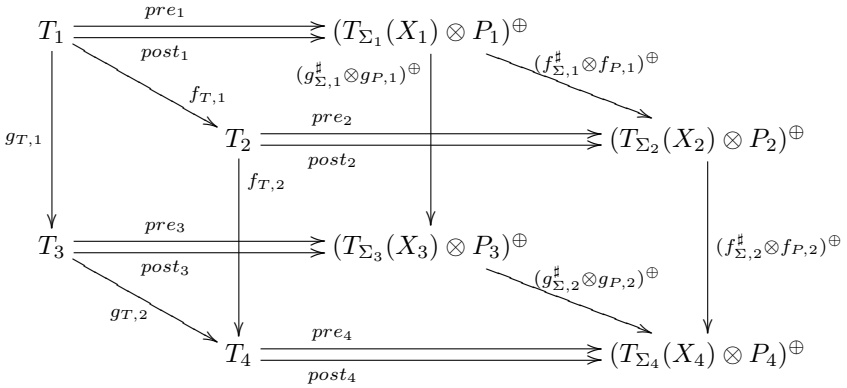
Due to the pushout property of $T_4$ there exists one and only one morphism

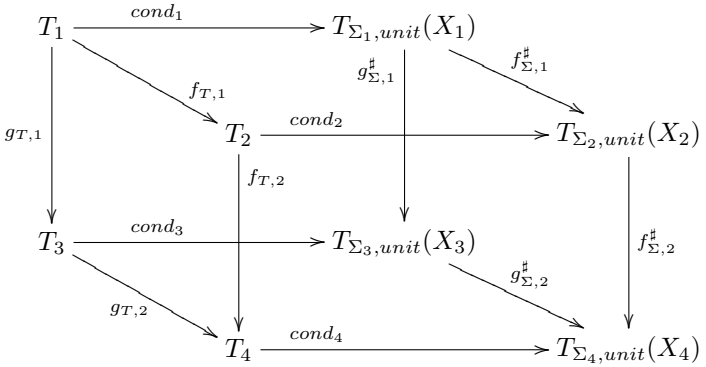
$$pre_4 : T_4 \longrightarrow (T_{\Sigma_4}(X_4)\otimes P_4)^\oplus,$$

such that

$$(f^\sharp_{\Sigma,2}\otimes f_{P,2})^\oplus \circ pre_2 = pre_4 \circ f_{T,2} \text{ and}$$
$$(g^\sharp_{\Sigma,2}\otimes g_{P,2})^\oplus \circ pre_3 = pre_4 \circ g_{T,2}.$$

Analogously the pushout property of $T_4$ yields $post_4 : T_4 \longrightarrow (T_{\Sigma_4}(X_4)\otimes P_4)^\oplus$.






Because $f_{N,1}$ and $g_{N,1}$ are AHO-net scheme morphisms we have

$$f^\sharp_{\Sigma,2} \circ cond_2 \circ f_{T,1} = g^\sharp_{\Sigma,2} \circ cond_3 \circ g_{T,1}.$$

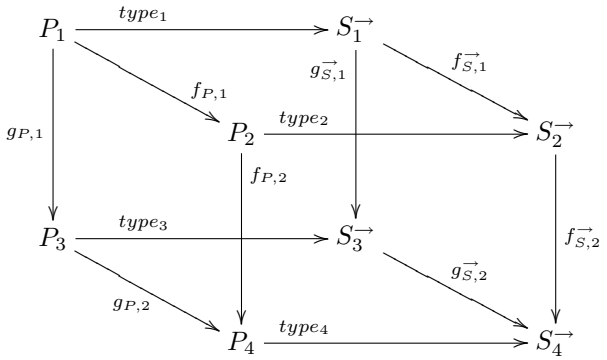
Thus, due to the pushout property of $T_4$, there exists one and only one morphism $cond_4 : T_4 \longrightarrow T_{\Sigma_4,unit}(X_4)$ such that

$$f^\sharp_{\Sigma,2} \circ cond_2 = cond_4 \circ f_{T,2} \text{ and}$$
$$g^\sharp_{\Sigma,2} \circ cond_3 = cond_4 \circ g_{T,2}.$$

Finally, because $f_{N,1}$ and $g_{N,1}$ are AHO-net scheme morphisms we have

$$f^{\rightarrow}_{S,2} \circ type_2 \circ f_{P,1} = g^{\rightarrow}_{S,2} \circ type_3 \circ g_{P,1}.$$

Due to the pushout property of $P_4$, there exists one and only one morphism $type_4 : P_4 \longrightarrow S^{\rightarrow}_4$ such that $f^{\rightarrow}_{S,2} \circ type_2 = type_4 \circ f_{P,2}$ and $g^{\rightarrow}_{S,2} \circ type_3 = type_4 \circ g_{P,2}$.




$f_{N,2} = (f_{\Sigma,2}, f_{P,2}, f_{T,2})$ and $g_{N,2} = (g_{\Sigma,2}, g_{P,2}, g_{T,2})$ are well-defined, i.e. they satisfy the compatibility of pre- and post domain functions, firing condition function and type function due to the induced morphisms $pre_4, post_4, cond_4$ and $type_4$. The commutativity follows directly from the componentwise construction of the AHO-net scheme morphisms and the commutativity of the diagrams above, i.e.

$$(g_{\Sigma,2} \circ g_{\Sigma,1}, g_{P,2} \circ g_{P,1}, g_{T,2} \circ g_{T,1}) = (f_{\Sigma,2} \circ f_{\Sigma,1}, f_{P,2} \circ f_{P,1}, f_{T,2} \circ f_{T,2})$$

implies

$$g_{N,2} \circ g_{N,1} = f_{N,2} \circ f_{N,1}.$$

$$\begin{array}{ccc} N_1 & \xrightarrow{f_{N,1}} & N_2 \\ {\scriptstyle g_{N,1}}\downarrow & = & \downarrow{\scriptstyle f_{N,2}} \\ N_3 & \xrightarrow[g_{N,2}]{} & N_4 \end{array}$$

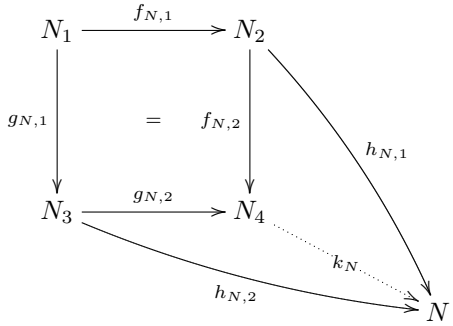
Next we check the pushout property. Given another AHO-net scheme

$$N = (\Sigma, P, T, pre, post, cond, type)$$

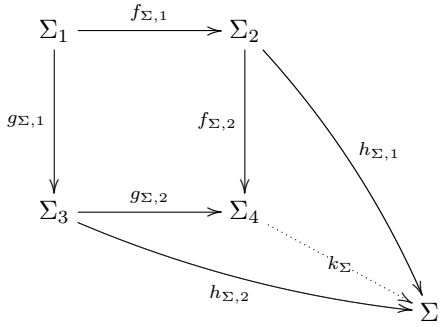
and a pair of AHO-net scheme morphisms

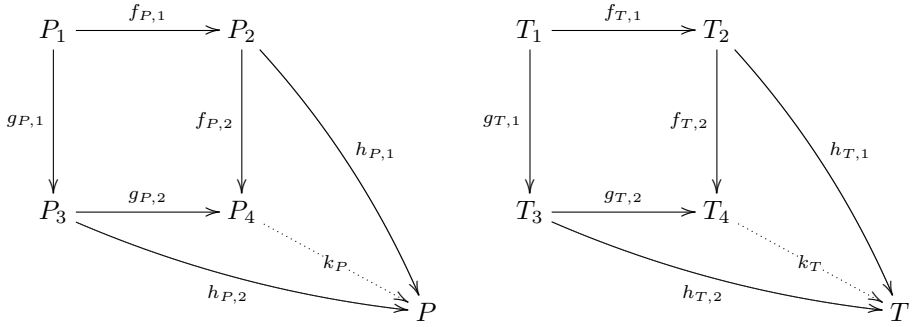
$$h_{N,1} = (h_{\Sigma,1}, h_{P,1}, h_{T,1}) : N_2 \longrightarrow N \text{ and}$$
$$h_{N,2} = (h_{\Sigma,2}, h_{P,2}, h_{T,2}) : N_3 \longrightarrow N,$$

such that $h_{N,1} \circ f_{N,1} = h_{N,2} \circ g_{N,1}$.




Then the induced morphism $k_N = (k_\Sigma, k_P, k_T) : N_4 \longrightarrow N$ is obtained by the induced morphisms $k_\Sigma, k_P$ and $k_T$ of the components $\Sigma_4, P_4$ and $T_4$.




Due to the construction of the pushout object $T_4$ in **Sets** we have for all $t_4 \in T_4$ :

$$\exists t_2 \in T_2 : t_4 = f_{T,2}(t_2) \text{ or } \exists t_3 \in T_3 : t_4 = g_{T,2}(t_3).$$

Let $t_4 \in T_4$ and $t_2 \in T_2$, such $t_4 = f_{T,2}(t_2)$.

$$\begin{aligned} pre \circ k_T(t_4) = &\quad pre \circ k_T \circ f_{T,2}(t_2) \\ = &\quad pre \circ h_{T,1}(t_2) \\ = &\quad (h^\sharp_{\Sigma,1} \otimes h_{P,1})^\oplus \circ pre_2(t_2) \\ = &\quad (k^\sharp_\Sigma \otimes k_P)^\oplus \circ (f^\sharp_{\Sigma,2} \otimes f_{P,2})^\oplus \circ pre_2(t_2) \\ = &\quad (k^\sharp_\Sigma \otimes k_P)^\oplus \circ pre_4 \circ f_{T,2}(t_2) \\ = &\quad (k^\sharp_\Sigma \otimes k_P)^\oplus \circ pre_4(t_4) \end{aligned}$$

and analogously for all $t_4 \in T_4$ and $t_3 \in T_3$ such that $t_4 = f_{T,3}(t_3)$. Analogously we can show

$$\begin{aligned} post \circ k_T &= (k^\sharp_\Sigma \otimes k_P)^\oplus \circ post_4, \\ cond \circ k_T &= k^\sharp_\Sigma \circ cond_4, \text{ and} \\ type \circ k_P &= k^{\rightarrow}_S \circ type_4 \end{aligned}$$

where $k_{S^\rightarrow} : S_4^\rightarrow \longrightarrow S^\rightarrow$ is induced by $k_S : S_4 \longrightarrow S$, $k^\sharp_\Sigma : T_{\Sigma_4}(X_4) \longrightarrow T_\Sigma(X)$ is the translation of $\Sigma_4$-terms by $k_\Sigma$ to $\Sigma$-terms and $(k^\sharp_\Sigma \otimes k_P)^\oplus$ is the unique homomorphic extension of $(k^\sharp_\Sigma \otimes k_P)$. Thus, $k$ is well-defined, i.e. they satisfy the compatibility of pre- and post domain functions, firing condition function and type function, i.e. the following diagrams commute.

$$\begin{array}{ccccc} T_{\Sigma_4,unit}(X_4) & \xleftarrow{cond_4} & T_4 & \underset{post_4}{\overset{pre_4}{\rightrightarrows}} & (T_{\Sigma_4}(X_4) \otimes P_4)^\oplus \\ \downarrow{\scriptstyle k^\sharp_\Sigma} & = & \downarrow{\scriptstyle k_T} & = & \downarrow{\scriptstyle (k^\sharp_\Sigma \otimes k_P)^\oplus} \\ T_{\Sigma,unit}(X) & \xleftarrow[cond]{} & T & \underset{post}{\overset{pre}{\rightrightarrows}} & (T_\Sigma(X) \otimes P)^\oplus \end{array}$$

$$\begin{array}{ccc} P_4 & \xrightarrow{type_4} & S_4^\rightarrow \\ \downarrow{\scriptstyle k_P} & = & \downarrow{\scriptstyle k_{S^\rightarrow}} \\ P & \xrightarrow[type]{} & S^\rightarrow \end{array}$$

Finally, uniqueness of $k$ is obtained by the uniqueness of $k_\Sigma, k_P$, and $k_T$. □

For the horizontal structuring technique fusion we investigate the (finite) colimits in the category of algebraic higher-order net schemes, i.e. we have to show, that the category **AHOS** permits an initial object, while the existence of all pushouts is proven above. Note that due to the finite cocompleteness the category **AHOS** has not only an initial object and all pushouts but also coequalizers (see Def. A.1.5 in the Appendix).

**Theorem 4.1.4 (Fusion)**
The fusion of two AHO-net scheme morphisms $f_N, g_N : F \longrightarrow N$ between two AHO-net schemes $F$ and $N$ is an AHO-net scheme $N'$ together with an AHO-net scheme morphism $h_N : N \longrightarrow N'$ defined by the coequalizer $h_N : N \longrightarrow N'$ of $f_N$ and $g_N$.

**Proof:** We have to show that **AHOS** has an initial object. Then we obtain due to Thm. 4.1.3 that **AHOS** is finitely cocomplete and also has all coequilizers. The initial object $N_\emptyset$ is given by

$$N_\emptyset = (\Sigma_\emptyset, P_\emptyset, T_\emptyset, pre_\emptyset, post_\emptyset, cond_\emptyset, type_\emptyset)$$

with

- $P_\emptyset = T_\emptyset = \emptyset$,
- $\Sigma_\emptyset$ the initial object in **HOSig** (see Fact 3.1.13), and
- $pre_\emptyset = post_\emptyset = cond_\emptyset = type_\emptyset$ the empty function.

Given an AHO-net $N = (\Sigma, P, T, pre, post, cond, type)$, then the unique AHO-net scheme morphism $f_{N,\emptyset} = (f_{\Sigma,\emptyset}, f_{P,\emptyset}, f_{T,\emptyset}) : N_\emptyset \longrightarrow N$ is given by

- $f_{P,\emptyset} = f_{T,\emptyset}$ the empty function and
- $f_{\Sigma,\emptyset}$ is given by the initiality of $\Sigma_\emptyset$ (see Fact 3.1.13),

where the compatibility of the pre- and post domain functions, the firing condition functions resp. the type functions holds due to the emptiness of $T_\emptyset$ and $P_\emptyset$. □

## 4.2 Structure Preserving Morphisms of AHO-Nets

Because AHO-nets are AHO-net schemes with a suitable model, morphisms of AHO-nets are a combination of AHO-net scheme morphisms and higher-order homomorphisms. Note that AHO-nets are defined with respect to a specific higher-order signature. So, here we use a restricted version of AHO-net scheme morphisms where the higher-order signature remains to be unchanged. Thus, the net inscriptions in the environment of transitions, which are related by the AHO-net morphism, are still the same. This will be important to achieve the preservation of the operational behavior by AHO-net morphisms.

**Definition 4.2.1 (Algebraic Higher-Order Net Morphisms)**
Given two AHO-nets $(N_i, A_i)$ wrt. the higher-order signature $\Sigma = (S, OP, X)$ with $N_i = (\Sigma, P_i, T_i, pre_i, post_i, cond_i, type_i)$ for $i \in \{1, 2\}$. Then an AHO-net morphism $f_{(N,A)} : (N_1, A_1) \longrightarrow (N_2, A_2)$ is given by

$$f_{(N,A)} = (f_N, f_A)$$

with

- $f_N = (id_\Sigma, f_P, f_T) : N_1 \longrightarrow N_2$ is an AHO-net scheme morphism (see Def. 4.1.1) with the identity higher-order signature morphism $id_\Sigma$ (see Fact 3.1.12),
- $f_A : A_1 \longrightarrow A_2$ is a higher-order homomorphism of $\Sigma$-algebras (see Def. 3.1.23).

Now, we define for each higher-order signature $\Sigma$ the category **AHON($\Sigma$)** of AHO-nets and AHO-net morphisms, where the data type part of AHO-nets consists of the higher-order signature $\Sigma$ together with a higher-order $\Sigma$-algebra.

**Fact 4.2.2 (Category AHON($\Sigma$))**
The category **AHON($\Sigma$)** consists of AHO-nets wrt. the higher-order signature $\Sigma$ as objects and AHO-net morphisms as morphisms.

**Proof:** We show that the composition defined componentwise is associative and the identity law is satisfied.

**Composition:** Given AHO-nets $(N_i, A_i)$ for $i \in \{1, 2, 3\}$) wrt. the higher-order signature $\Sigma$ and a pair of AHO-net morphisms

$$f_{(N,A),1} : (N_1, A_1) \longrightarrow (N_2, A_2) \text{ and } f_{(N,A),2} : (N_2, A_2) \longrightarrow (N_3, A_3),$$

the composition of $f_{(N,A),1}$ and $f_{(N,A),2}$ defined componentwise by

$$f_{(N,A),2} \circ f_{(N,A),1} = (f_{N,2} \circ f_{N,1}, f_{A,2} \circ f_{A,1}) : (N_1, A_1) \longrightarrow (N_3, A_3)$$

is an AHO-net morphism because AHO-net scheme morphisms are closed under composition (see Fact 4.1.2) and higher-order homomorphisms are closed under composition (see Fact 3.1.25).

**Associativity:** Let $f_{(N,A),i} = (f_{N_i}, f_{A_i})$ for $i \in \{1,2,3\}$ be three AHO-net morphisms. Then the composition is associative because it is defined componentwise, i.e. we have

$$(f_{(N,A),3} \circ f_{(N,A),2}) \circ f_{(N,A),1} = f_{(N,A),3} \circ (f_{(N,A),2} \circ f_{(N,A),1}).$$

**Identity:** The identity is given by the identity AHO-net morphisms. Because the composition is defined componentwise, the identity law is satisfied, i.e. we have for $f_{(N,A)} = (f_N, f_A) : (N, A) \longrightarrow (N', A')$

$$f_{(N,A)} \circ id_{(N,A)} = f_{(N,A)} \text{ and } id_{N'} \circ f_N = f_N.$$

□

In the following we show that AHO-net morphisms preserve the firing behavior. In more detail, if there is a transition and a variable valuation in the source net, so that the net inscriptions in the environment of the transitions are defined, then there is a consistent transition assignment in the target net, which consists of the image of the transition, and the composition of the variable valuation and the higher-order homomorphism given by the AHO-net morphism. Moreover, not only the marking in the source net is mapped to a marking in the target net, but also the follower marking computed by the firing of the transition in the source net is mapped to the follower marking computed by the firing of image of the transition. Here we frequently use on the one hand the fact that both the transition and the image of the transition have the same net inscriptions in their environment. On the other hand we make use of the definedness condition of higher-order homomorphisms to ensure that the evaluation of the net inscriptions is defined in the target net as well.

**Theorem 4.2.3 (AHO-Net Morphisms Preserve Firing Behavior)**
Let $(N_i, A_i)$ for $i \in \{1,2\}$ wrt. the higher-order signature $\Sigma$ be two AHO-nets with

$$N_i = (\Sigma, P_i, T_i, pre_i, post_i, cond_i, type_i)$$

and let

$$f_{(N,A)} = (f_N, f_A) : (N_1, A_1) \longrightarrow (N_2, A_2) \text{ with } f_N = (id_\Sigma, f_P, f_T)$$

be an AHO-net morphism. Moreover, let $M \in (A_1 \otimes P_1)^\oplus$ be a marking of $(N_1, A_1)$ and $(t, v) \in CT_1$ a consistent transition valuation such that $t$ is enabled in $M \in CP_1^\oplus$ under $v$ and let $M' \in CP_1^\oplus$ be the follower marking of $M$ after firing of $t$. Then we have

1. a consistent transition valuation $(f_T(t), f_A \circ v) \in CT_2$,
2. $f_T(t)$ is enabled in the marking $(f_A \otimes f_P)^\oplus(M) \in CP_2^\oplus$ under $(f_A \circ v)$, and
3. the firing of the transition $f_T(t)$ results in the follower marking

$$(f_A \otimes f_P)^\oplus(M') \in CP_2^\oplus$$

that is

$$M[(t,v)\rangle M' \Longrightarrow (f_A \otimes f_P)^{\oplus}(M)[(f_T(t), f_A \circ v)\rangle (f_A \otimes f_P)^{\oplus}(M'),$$

where $(f_A \otimes f_P)$ denotes the corresponding function of type consisting markings defined for all $(a,p) \in (A_1 \otimes P_1)$ by $(f_A \otimes f_P)(a,p) = (f_A(a), f_P(p))$ and $(f_A \otimes f_P)^{\oplus}$ is the unique homomorphic extension of $(f_A \otimes f_P)$ (see Def. B.1.2 in the Appendix).

**Proof:** First note that due to the definition of AHO-net morphisms a transition $t \in T_1$ and its image $f_T(t) \in T_2$ have the same net inscriptions in their environment, i.e. we have for all $t \in T_1$:

$$cond_2 \circ f_T(t) = id_{\Sigma}^{\sharp} \circ cond_1(t) = cond_1(t), \tag{4.1}$$

$$pre_2(f_T(t)) = (id_{\Sigma}^{\sharp} \otimes f_P)^{\oplus}(pre_1(t)), \tag{4.2}$$

and analogously for the post domain. Thus, we have for all $t \in T_1$:

$$(term, p_1) \in pre_1(t) \Longrightarrow (term, f_P(p_1)) \in pre_2(f_T(t)), \tag{4.3}$$

$$\bullet(f_T(t)) = f_P(\bullet t), \tag{4.4}$$

and analogously for the post domain.

1. It remains to show that $(f_T(t), f_A \circ v)$ is a consistent transition valuation, i.e. for all $t \in T$ we have

   $$cond_2(f_T(t)) \in dom((f_A \circ v)^{\sharp})$$

   and

   $$\forall (term, p_2) \in pre_2(f_T(t)) \oplus post_2(f_T(t)) : term \in dom((f_A \circ v)^{\sharp}).$$

   Because $(t,v) \in CT_1$ we have

   $$\begin{array}{lll} & cond_1(t) \in dom(v^{\sharp}) & \\ \Longrightarrow & cond_1(t) \in dom(f_A \circ v^{\sharp}) & \text{(Fact 3.1.24)} \\ \Longrightarrow & cond_1(t) \in dom((f_A \circ v)^{\sharp}) & \text{(Fact 3.1.24)} \\ \Longrightarrow & cond_2(f_T(t)) \in dom((f_A \circ v)^{\sharp}) & \text{(by (4.1))} \end{array}$$

   and

   $$\begin{array}{ll} & \forall (term, p_1) \in pre_1(t) \oplus post_1(t) : term \in dom(v^{\sharp}) \\ \Longrightarrow & \forall (term, p_1) \in pre_1(t) \oplus post_1(t) : term \in dom(f_A \circ v^{\sharp}) \\ & \qquad \text{(Fact 3.1.24)} \\ \Longrightarrow & \forall (term, p_1) \in pre_1(t) \oplus post_1(t) : term \in dom((f_A \circ v)^{\sharp}) \\ & \qquad \text{(Fact 3.1.24)} \\ \Longrightarrow & \forall (term, f_P(p_1)) \in pre_2(f_T(t)) \oplus post_2(f_T(t)) : term \in dom((f_A \circ v)^{\sharp}) \\ & \qquad \text{(by (4.3))} \\ \Longrightarrow & \forall (term, p_2) \in pre_2(f_T(t)) \oplus post_2(f_T(t)) : term \in dom((f_A \circ v)^{\sharp}) \\ & \qquad \text{(by (4.4))}. \end{array}$$

   It follows that $(f_T(t), f_A \circ v)$ is a consistent transition valuation.

2. Because $t$ is enabled in $M \in (A_1 \otimes P_1)^{\oplus}$ under $v$ we have

$$\begin{array}{ll} & \hat{v}(pre_1(t)) \leq M \\ \Longrightarrow & (f_A \otimes f_P)^{\oplus}(\hat{v}(pre_1(t))) \leq (f_A \otimes f_P)^{\oplus}(M) \\ \Longrightarrow & (\widehat{f_A \circ v})(pre_2(f_T(t)) \leq (f_A \otimes f_P)^{\oplus}(M) \qquad \text{(see Proof of } \mathbf{3.}\text{)}. \end{array}$$

It follows that $f_T(t)$ is enabled in $(f_A \otimes f_P)^{\oplus}(M)$ under $(f_A \circ v)$.

3. Formally we have to show:

$$\begin{array}{rcl} M' & = & M \ominus \hat{v}(pre_1(t)) \oplus \hat{v}(post_1(t)) \\ \Longrightarrow \quad (f_A \otimes f_P)^{\oplus}(M') & = & (f_A \otimes f_P)^{\oplus}(M) \ominus (\widehat{f_A \circ v})(pre_2(f_T(t))) \\ & & \oplus (\widehat{f_A \circ v})(post_2(f_T(t))). \end{array}$$

Because $(f_A \otimes f_P)^{\oplus}$ is a monoid homomorphism we get

$$\begin{array}{ll} & (f_A \otimes f_P)^{\oplus}(M') \\ = & (f_A \otimes f_P)^{\oplus}(M \ominus \hat{v}(pre_1(t)) \oplus \hat{v}(post_1(t))) \\ = & (f_A \otimes f_P)^{\oplus}(M) \ominus (f_A \otimes f_P)^{\oplus}(\hat{v}(pre_1(t))) \oplus (f_A \otimes f_P)^{\oplus}(\hat{v}(post_1(t))). \end{array}$$

So we have to show that the following equations hold:

$$\begin{array}{l} (f_A \otimes f_P)^{\oplus}(\hat{v}(pre_1(t))) = (\widehat{f_A \circ v})(pre_2(f_T(t))) \text{ and} \\ (f_A \otimes f_P)^{\oplus}(\hat{v}(post_1(t))) = (\widehat{f_A \circ v})(post_2(f_T(t))). \end{array}$$

Let $pre_1(t) = \sum_{i=1}^{n}(term_i, p_i)$, then

$$\begin{array}{lll} & (f_A \otimes f_P)^{\oplus}(\hat{v}(pre_1(t))) & \\ = & (f_A \otimes f_P)^{\oplus}(\hat{v}(\sum_{i=1}^{n}(term_i, p_i))) & \\ = & (f_A \otimes f_P)^{\oplus}(\sum_{i=1}^{n}(v^{\sharp}(term_i), p_i)) & \\ = & \sum_{i=1}^{n}((f_A \circ v^{\sharp})(term_i), f_P(p_i)) & \\ = & \sum_{i=1}^{n}((f_A \circ v)^{\sharp}(term_i), f_P(p_i)) & \text{(Fact 3.1.24)} \\ = & (\widehat{f_A \circ v})(\sum_{i=1}^{n}(term_i, f_P(p_i))) & \\ = & (\widehat{f_A \circ v})((id_{\Sigma}^{\sharp} \otimes f_P)^{\oplus}(\sum_{i=1}^{n}(term_i, p_i))) & \\ = & (\widehat{f_A \circ v})((id_{\Sigma}^{\sharp} \otimes f_P)^{\oplus}(pre_1(t))) & \\ = & (\widehat{f_A \circ v})(pre_2(f_T(t))) & \text{(by (4.2))} \end{array}$$

and analogously for the post domain. It follows that the firing of the transition $f_T(t)$ results in the follower marking $(f_A \otimes f_P)^{\oplus}(M')$.

□

## 4.3 Process Semantics

In this section we introduce the notion of higher-order processes based on a suitable notion of higher-order occurrence nets. The main idea is to use the same properties

like unitary, conflict freeness and acyclicity as in the low-level case, but drop the idea that an occurrence net captures essentially one concurrent computation. In the low-level case the markings of the input places are given by indistinguishable black tokens, while in the higher-order case they are rather complex data elements. Thus, the input parameters can be given by different data elements leading to different concurrent computations. Because AHO-nets are closely related to algebraic high-level nets, the following definitions adopt the notions obtained for high-level processes in [EHP+02]. The main differences between these notions are on the one hand that the data type part of a higher-order occurrence net is given by a higher-order signature and corresponding higher-order partial algebra instead of a classical algebraic specification and corresponding classical algebra, and on the other hand that the firing conditions are defined by atomic formulas instead of a finite set of equations.

**Definition 4.3.1 (Algebraic Higher-Order Occurrence Net)**
A (deterministic) algebraic higher-order occurrence net $(K, A)$ wrt. the higher-order signature $\Sigma$ is an AHO-net with

$$K = (\Sigma, P, T, pre, post, cond, type)$$

such that for all $t \in T$ with $pre(t) = \sum_{i=1}^{n}(term_i, p_i)$ we have

1. (*Unarity*): for all $t \in T$ and for all $p \in \bullet t$ the arc from $p$ to $t$ has a unary arc-inscription $term$ rather than a proper sum of terms (see Remark 3.2.2), i.e. we have $pre(t)_{|p} = term$ and $p_1, \ldots, p_n$ are pairwise distinct (analogously for all $t \in T$ and $p \in t\bullet$).
2. (*No Forward Conflicts*): $\bullet t \cap \bullet t' = \emptyset$ for all $t, t' \in T, t \neq t'$
3. (*No Backward Conflicts*): $t\bullet \cap t'\bullet = \emptyset$ for all $t, t' \in T, t \neq t'$
4. (*Partial Order*): the causal relation $<\subseteq (P \times T) \cup (T \times P)$ defined by the transitive closure of

$$\{(p, t) \in P \times T \mid p \in \bullet t\} \cup \{(t, p) \in T \times P \mid p \in t\bullet\}$$

is a finitary strict partial order, i.e. the partial order is irreflexive and for each element in the partial order the set of its predecessors is finite.

An algebraic higher-order process of an AHO-net is an AHO-net morphism such that the source net is an AHO-occurrence net and the data type part is preserved.

**Definition 4.3.2 (Algebraic Higher-Order Process)**
A (deterministic) algebraic higher-order process of an AHO-net $(N, A)$ wrt. the higher-order signature $\Sigma$ is an AHO-net morphism

$$p = (p_N, id_A) : (K, A) \longrightarrow (N, A) \text{ with } p_N = (id_\Sigma, f_P, f_T),$$

where $(K, A)$ is a (deterministic) AHO-occurrence net wrt. the higher-order signature $\Sigma$ with the same higher-order partial algebra $A$, i.e. the data type part is preserved by $p$.

In this thesis we only define deterministic AHO-occurrence nets leading to deterministic AHO-processes. The nondeterministic case can be obtained by dropping the second condition "No forward conflicts" because conflicts are allowed in nondeterministic processes. If we drop the first condition "Unarity", we achieve the notion of AHO-multi-occurrence nets leading to AHO-multi-processes. Further investigations of these different notions of higher-order processes will be a part of future work.

# Chapter 5

# Folding and Unfolding Techniques

In this chapter we provide folding and unfolding constructions, which are new in the area of Petri nets. In connection with our striving goals we present folding constructions wrt. constant symbols in Section 5.1. The main idea is to replace some parts of the net inscriptions by formal parameters, more precisely specific constants are substituted by appropriate variables. A special case of this folding construction occurs in form of constant symbols of function types. Note that in our higher-order setting operation symbols can also be formulated as constant symbols together with application symbols of appropriate types. In this way, the folding construction realizes a high-level of abstraction, because the operations are not fixed in the net structure, but can be given at run time by suitable tokens. From a practical point of view these folding constructions support the demanded flexibility and adaptability of models by operation late-binding mechanisms. Furthermore, changes in the environment result in an exchange of tokens instead of a modification of the net structure. We show that we can give a folding construction wrt. constant symbols, so that the firing behavior is preserved. Moreover, we give an unfolding construction wrt. constant symbols preserving the operational behavior. The unfolding construction has got a counterpart in the context of high-level nets, where the flattening construction is used to define the semantics of a high-level net by a classical Petri net (see e.g. [Jen92, Hum89, Gen86, Pad96]). Vice versa the concept of folding construction is related to the step from low to high-level nets. An important aspect is that the folding and unfolding constructions are inverse to each other, so that we can switch between the levels of abstractions.

In Section 5.2 we exploit the fact that in our approach of AHO-nets product types are available. We can give a folding construction wrt. product types leading to more compact description of models, if the given net includes a specific net structure, called set of uniform places. Then the set of uniform places is merged into one place of suitable product type and the particular net inscriptions are formed by tuples. The main results show that the folding construction wrt. product types preserves the operational behavior and that we can give an inverse construction, called unfolding wrt. product types, so that the firing behavior is preserved. Both folding constructions wrt. constant symbols and folding constructions wrt. product types are extensively used in the case study “Logistics” in Chapter 9.

## 5.1 Folding and Unfolding wrt. Constant Symbols

In this section we give the main results of the folding and unfolding constructions wrt. constant symbols. Examples can be found in Section 3.3, where we apply a number of folding constructions to the AHO-net "Computation II" (see Fig. 3.2) to obtain the more abstract representation of the process given by the AHO-net "Computation IV" in Fig. 3.4. The folding and unfolding constructions wrt. constant symbols are first presented by the transformation of terms and then by the folding and unfolding of the net part. Note that we do not transform the data type part, i.e. we assume that there are suitable constant symbols with corresponding variables given in the higher-order signature. The advantage is that the composition of folding and unfolding constructions leads to equivalent data type parts. Otherwise we have to add several variables (one for each application of the folding construction) and several constant symbols (one for each application of the unfolding construction). But we are not able to remove some of these constant symbols and variables because they might be used in the environment of other transitions than the (un-)folded one.

We define a fold-operator to transform terms of a higher-order signature $\Sigma$ by replacing a specific constant symbol by a corresponding variable. For instance the term $(add.\langle x, y\rangle)$ of the signature HO-NAT in Section 3.3 is transformed into the term $(f_{add}.\langle x, y\rangle)$, where the variable $f_{add}$ is assigned to the function type $(Nat \star Nat \to Nat)$.

**Definition 5.1.1 (Folding of Terms)**
Let $\Sigma = (S, OP, X)$ be a higher-order signature, $c : type \in OP$ a constant symbol and $f_c : type \in X$ a variable, the fold-operator $fold_c : T_\Sigma(X) \longrightarrow T_\Sigma(X)$ is recursively defined by

**1.** $fold_c(\langle\rangle) = \langle\rangle$ for $\langle\rangle \in T_{\Sigma,unit}(X)$

**2.** $fold_c(x) = x$ for all $x : type \in X$

**3a.** $fold_c(op) = f_c$ for $op : type \in OP$ and $op = c$

**3b.** $fold_c(op(term)) = op(fold_{op}(term))$
for all $op : type_1 \rightharpoonup type_2 \in OP, op \neq c,$ and $term \in T_{\Sigma,type_1}(X)$

**4.** $fold_{op}(\langle term_1, \ldots, term_n\rangle) = \langle fold_{op}(term_1), \ldots, fold_{op}(term_n)\rangle$
for all $term_1 \in T_{\Sigma,type_1}(X), \ldots, term_n \in T_{\Sigma,type_n}(X)$

**5.** $fold_c(pr_i^{(type_1 \star \ldots \star type_n)}(term)) = pr_i^{(type_1 \star \ldots \star type_n)}(fold_c(term))$
for all $pr_i^{(type_1 \star \ldots \star type_n)} \in OP^{\rightharpoonup} \setminus OP$ and $term \in T_{\Sigma,(type_1 \star \ldots \star type_n)}(X)$.

Analogously, we define an unfold-operator to transform terms of a higher-order signatures by replacing a specific variable by a corresponding constant symbol. In terms of our example in Section 3.3 the application of the unfold-operator transforms the term $(f_{add}.\langle x, y\rangle)$ into the term $(add.\langle x, y\rangle)$.

**Definition 5.1.2 (Unfolding of Terms)**
Let $\Sigma = (S, OP, X)$ be a higher-order signature, $c : type \in OP$ a constant symbol and $f_c : type \in X$ a variable, the unfold-operator $unfold_{f_c} : T_\Sigma(X) \longrightarrow T_\Sigma(X)$ is

recursively defined by

**1.** $unfold_{f_c}(\langle\rangle) = \langle\rangle$ for $\langle\rangle \in T_{\Sigma,unit}(X)$

**2a.** $unfold_{f_c}(x) = c$ for $x \in X$ and $x = f_c$

**2b.** $unfold_{f_c}(x) = x$ for all $x \in X$ and $x \neq f_c$

**3.** $unfold_{f_c}(op(term)) = op(unfold_{f_c}(term))$
for all $op : type_1 \rightharpoonup type_2 \in OP,$ and $term \in T_{\Sigma,type_1}(X)$

**4.** $unfold_{f_c}(\langle term_1, \ldots, term_n\rangle) = \langle unfold_{f_c}(term_1), \ldots, unfold_{f_c}(term_n)\rangle$
for all $term_1 \in T_{\Sigma,type_1}(X), \ldots, term_n \in T_{\Sigma,type_n}(X)$

**5.** $unfold_{f_c}(pr_i^{(type_1 \star \ldots \star type_n)}(term)) = pr_i^{(type_1 \star \ldots \star type_n)}(unfold_{f_c}(term))$
for all $pr_i^{(type_1 \star \ldots \star type_n)} \in OP^{\rightarrow} \setminus OP$ and $term \in T_{\Sigma,(type_1 \star \ldots \star type_n)}(X)$.

In the following we show that the composition of the fold-operator and the unfold-operator defined above leads to the same term, which is essential for the result that the folding and unfolding constructions are inverse. Note that we have to restrict the input parameter of the composition.

**Lemma 5.1.3 (Folding- and Unfolding-Operators are inverse)**
Given a higher-order signature $\Sigma = (S, OP, X)$, a constant symbol $c : type \in OP$, and a variable $f_c : type \in X$, then we have

$$unfold_{f_c}(fold_c(term)) = term$$

if $f_c \notin Var(term)$ and

$$fold_c(unfold_{f_c}(term)) = term$$

if $c \notin Op(term)$.

**Proof:** First we show that $unfold_{f_c}(fold_c(term)) = term$. By induction over the term structure of $T_\Sigma(X)$ we have:

**Induction Base:**

**1.** $unfold_{f_c}(fold_c(\langle\rangle)) = unfold_{f_c}(\langle\rangle) = \langle\rangle$ for $\langle\rangle \in T_{\Sigma,unit}(X)$

**2** $unfold_{f_c}(fold_c(x)) = unfold_{f_c}(x) = x$ for all $x : type \in X$ and $x \neq f_c$.

**Induction Step:** Let $unfold_{f_c}(fold_c(term_i)) = term_i$ for some terms $term_i \in T_\Sigma(X)$ for $i \in \{1, \ldots, n\}$.

**3a.** $unfold_{f_c}(fold_c(op)) = unfold_{f_c}(fold_c(c)) = unfold_{f_c}(f_c) = c = op$

for $op : type \in OP$ and $op = c$

**3b.**

$$\begin{aligned} unfold_{f_c}(fold_c(op(term))) &= unfold_{f_c}(op(fold_c(term))) \\ &= op(unfold_{f_c}(fold_c(term)) \\ &= op(term) \end{aligned}$$

for all $op : type_1 \rightharpoonup type_2 \in OP, op \neq c,$ and all $term \in T_{\Sigma,type_1}(X)$

**4.**

$$\begin{aligned} & unfold_{f_c}(fold_c(\langle term_1, \ldots, term_n \rangle)) \\ = \; & unfold_{f_c}(\langle fold_c(term_1), \ldots, fold_c(term_n) \rangle) \\ = \; & \langle unfold_{f_c}(fold_c(term_1)), \ldots, unfold_{f_c}(fold_c(term_n)) \rangle \\ = \; & \langle term_1, \ldots, term_n \rangle \end{aligned}$$

for all $term_1 \in T_{\Sigma,type_1}(X), \ldots, term_n \in T_{\Sigma,type_n}(X)$

**5.**

$$\begin{aligned} & unfold_{f_c}(fold_c(pr_i^{(type_1 \star \ldots \star type_n)}(term))) \\ = \; & unfold_{f_c}(pr_i^{(type_1 \star \ldots \star type_n)}(fold_c(term))) \\ = \; & pr_i^{(type_1 \star \ldots \star type_n)}(unfold_{f_c}(fold_c(term))) \\ = \; & pr_i^{(type_1 \star \ldots \star type_n)}(term) \end{aligned}$$

for all $pr_i^{(type_1 \star \ldots \star type_n)} \in OP^{\rightarrow} \setminus OP$ and $term \in T_{\Sigma,(type_1 \star \ldots \star type_n)}(X)$.

The proof of $fold_c(unfold_{f_c}(term)) = term$ is more or less analog. □

Next we give the folding construction of an AHO-net scheme wrt. constant symbols. The folding construction is applied to a specific transition $t$, so that the constant symbol $c$ occurs in its environment, but we have to ensure that the corresponding variable $f_c$ does not occur in its net inscriptions. Otherwise the firing of this transition may result in a counter-intuitive behavior after the application of the folding construction. The data type part and the set of transitions remain to be unchanged. The set of places is given for the folded AHO-net by adding a contextual place (see Def. 3.2.4) with an appropriate type. Furthermore, we apply the fold-operator to the net inscriptions of the transition $t$. The set of places in the environment of the transition $t$ is given by adding the new contextual place with arc inscriptions given by the variable $f_c$. A general description of the folding construction wrt. constant symbols is depicted in Fig. 5.1.

**Fact 5.1.4 (Folding of AHO-net schemes wrt. constant symbols)**
Given an AHO-net scheme $N = (\Sigma, P, T, pre, post, cond, type)$ with $\Sigma = (S, OP, X)$, $c : type \in OP$ and $f_c : type \in X$. Let $t^c \in T$ with $f_c \notin Var(t^c)$ and $c \in Op(t^c)$, the folding of $N$ wrt. the constant symbol $c$ with corresponding variable $f_c$ and the transition $t^c$ is an AHO-net scheme

$$F(N) = (F(\Sigma), F(P), F(T), F(pre), F(post), F(cond), F(type))$$

defined by

- $F(\Sigma) = \Sigma$,
- $F(P) = P \uplus \{p^c\}$, where $p^c$ is a new place corresponding to the transition $t^c$ and the constant symbol $c$,
- $F(T) = T$,
- $F(pre) : T \longrightarrow (T_\Sigma(X) \otimes F(P))^{\oplus}$ with

$$F(pre)(t) = \begin{cases} \sum_{i=1}^{n}(fold_c(term_i), p_i) \oplus (f_c, p^c) \\ \qquad\qquad \text{if } t = t^c \text{ and } pre(t^c) = \sum_{i=1}^{n}(term_i, p_i) \\ pre(t) \qquad \text{else} \end{cases}$$

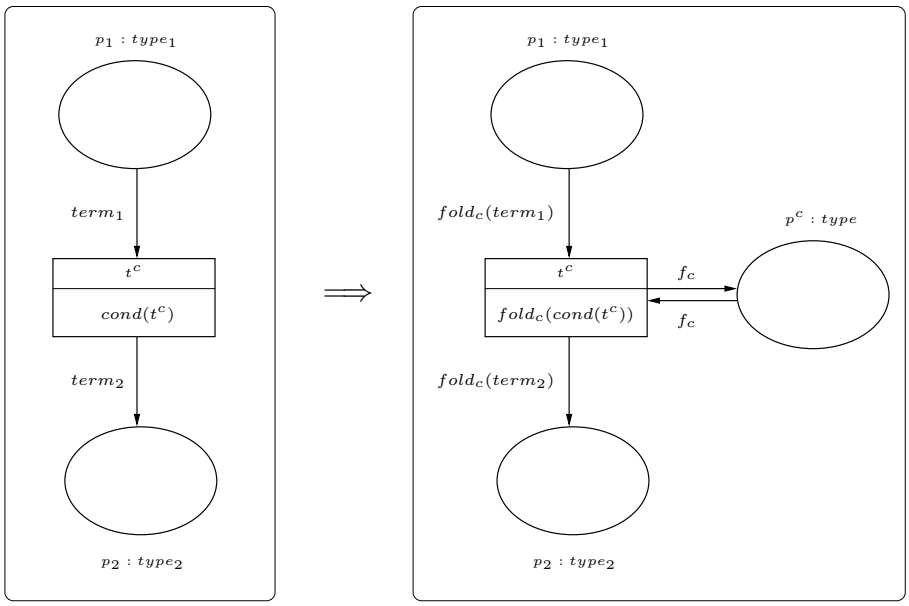



Figure 5.1: Schemata of folding construction wrt. constant symbols

and $F(post) : T \longrightarrow (T_\Sigma(X) \otimes F(P))^\oplus$ with

$$F(post)(t) = \begin{cases} \sum_{j=1}^{m}(fold_c(term_j), p_j) \oplus (f_c, p^c) \\ \qquad\qquad \text{if } t = t^c \text{ and } post(t^c) = \sum_{j=1}^{m}(term_j, p_j) \\ post(t) \qquad \text{else} \end{cases}$$

- $F(cond) : T \longrightarrow T_{\Sigma,unit}(X)$ with

$$F(cond)(t) = \begin{cases} fold_c(cond(t)) & \text{if } t = t^c \\ cond(t) & \text{else} \end{cases}$$

- $F(type) : F(P) \longrightarrow S^\rightarrow$ with

$$F(type)(p) = \begin{cases} type & \text{if } p = p^c \text{ and } c : type \\ type(p) & \text{else} \end{cases}$$

**Proof (Sketch):** We have to show that $F(N)$ is an AHO-net scheme as defined in Def. 3.2.1. For all $t \in T$ and $t \neq t^c$ the environment is preserved by the folding construction. The definition of the fold-operator (see Def. 5.1.1) guarantees that the net inscriptions and the firing condition of the folded transition $t^c$ consists of terms of the higher-order signature. Moreover, we have $(f_c, p^c) \in (T_\Sigma(X) \otimes F(P))^\oplus$ because the type assigned to the generated contextual place $p^c$ corresponds to the type of the constant symbol $c$ and thus to the type of the variable $f_c$. □

In our example in Section 3.3 we apply the folding construction wrt. the constant symbol *add* with corresponding variable $f_{add}$ and the transition *compute add* to the AHO-net "Computation II" (see Fig. 3.2). The resulting AHO-net "Computation III" is depicted in Fig. 3.3. Let the AHO-net "Computation II" be defined by $(N_{II}, HO\text{-}A)$ with $N_{II} = ((HO\text{-}Nat, X), P_{II}, T_{II}, pre_{II}, post_{II}, cond_{II}, type_{II})$.

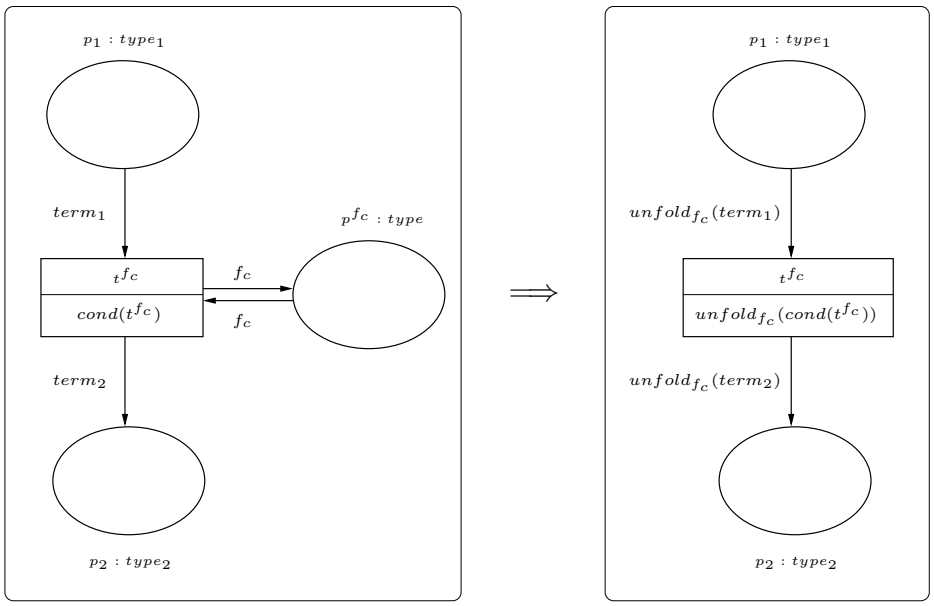



Figure 5.2: Schemata of unfolding construction wrt. constant symbols

Then the set of transitions and the higher-order signature remains unchanged in the AHO-net "Computation III". The set of places $F(P_{II})$ of the AHO-net "Computation III" is given by $F(P_{II}) = P_{II} \uplus \{p^{add}\}$. While the environment of the transition *compute sub* is not effected by the folding construction, the pre- and post domain and the firing condition of the transition *compute add* is defined by the following:

$$\begin{aligned}
F(pre_{II})(compute\ add) &= (fold_{add}(\langle x, y\rangle), p^F) \oplus (f_{add}, p^{add}) \\
&= (\langle x, y\rangle, p^F) \oplus (f_{add}, p^{add}), \\
F(post_{II})(compute\ add) &= (fold_{add}(add.\langle x, y\rangle), p_3) \oplus (f_{add}, p^{add}) \\
&= ((f_{add}.\langle x, y\rangle), p_3) \oplus (f_{add}, p^{add}), \text{ and} \\
F(cond_{II})(compute\ add) &= fold_{add}(leq.\langle x, y\rangle) \\
&= (leq.\langle x, y\rangle).
\end{aligned}$$

Finally, the types assigned to the places $p^F$ and $p_3$ are still the same. Furthermore, we have $type(p^{add}) = (Nat \star Nat \to Nat)$, because $add : Nat \star Nat \to Nat$.

Next we define the unfolding construction of an AHO-net scheme wrt. constant symbols. The unfolding construction is applied to a specific transition $t$, so that there is a contextual place in the environment of this transition with unary arc inscriptions given by a variable $f_c$. Moreover, we have to make sure that the constant symbol $c$, which corresponds to the variable $f_c$, does not occur in its net inscriptions because otherwise the firing behavior of the AHO-net and the unfolded AHO-net are not equivalent. The data type part and the set of transitions remain unchanged. The set of places of the folded AHO-net is given by deleting the contextual place. Furthermore, we apply the unfold-operator to the net inscriptions of the transition $t$. A general description of the folding construction wrt. constant symbols is depicted in Fig. 5.2.

**Fact 5.1.5 (Unfolding of AHO-net schemes wrt. constant symbols)**
Let $N = (\Sigma, P, T, pre, post, cond, type)$ be an AHO-net scheme with $\Sigma = (S, OP, X)$, $c : type \in OP$ and $f_c : type \in X$. Let $t^{f_c} \in T$ such that $c \notin Op(t^{f_c})$. Moreover, let $p^{f_c} \in P$ a contextual place (see Def. 3.2.4) such that

$$pre(t^{f_c})_{|p^{f_c}} = post(t^{f_c})_{|p^{f_c}} = f_c.$$

The unfolding of $N$ wrt. the constant symbol $c$ with corresponding variable $f_c$, the transition $t^{f_c}$ and the contextual place $p^{f_c}$ is an AHO-net scheme

$$U(N) = (U(\Sigma), U(P), U(T), U(pre), U(post), U(cond), U(type))$$

defined by

- $U(\Sigma) = \Sigma$,
- $U(P) = P \setminus \{p^{f_c}\}$,
- $U(T) = T$,
- $U(pre) : T \longrightarrow (T_\Sigma(X) \otimes U(P))^\oplus$ with

$$U(pre)(t) = \begin{cases} \sum_{i=1}^{n}(unfold_{f_c}(term_i), p_i) & \text{if } t = t^{f_c} \text{ and} \\ \qquad pre(t^{f_c}) = \sum_{i=1}^{n}(term_i, p_i) \oplus (f_c, p^{f_c}) & \\ pre(t) & \text{else} \end{cases}$$

  and $U(post) : T \longrightarrow (T_\Sigma(X) \otimes U(P))^\oplus$ with

$$U(post)(t) = \begin{cases} \sum_{j=1}^{m}(unfold_{f_c}(term_j), p_j) & \text{if } t = t^{f_c} \text{ and} \\ \qquad post(t^{f_c}) = \sum_{j=1}^{m}(term_j, p_j) \oplus (f_c, p^{f_c}) & \\ post(t) & \text{else} \end{cases}$$

- $U(cond) : T \longrightarrow T_{\Sigma,unit}(X)$ with

$$U(cond)(t) = \begin{cases} unfold_{f_c}(cond(t)) & \text{if } t = t^{f_c} \\ cond(t) & \text{else} \end{cases}$$

- $U(type) : U(P) \longrightarrow S^{\rightarrow}$ with $U(type)(p) = type(p)$.

**Proof (Sketch):** We have to show that $U(N)$ is an AHO-net scheme as defined in Def. 3.2.1. For all $t \in T$ and $t \neq t^{f_c}$ we have that the environment remains unchanged. The definition of the unfold-operator (see Def. 5.1.2) guarantees that the net inscriptions and the firing condition of the unfolded transition $t^c$ consist of terms of the higher-order signature. Finally, for all $p \in U(P)$ the assigned type is still the same as in $N$. □

In our example in Section 3.3 the place $p^{add}$ in AHO-net "Computation III" (see Fig. 3.3) is a contextual place because the connection between this place and the transition *compute add* is given by a double arrow labeled with the variable $f_{add}$. The application of the unfolding construction wrt. the constant symbol *add* with corresponding variable $f_{add}$ and the transition *compute add* with contextual place $p^{add}$ leads to the AHO-net "Computation II" depicted in Fig. 3.2.

Next we show that the composition of the folding and unfolding construction wrt. constant symbols leads to equivalent nets. This result is not only essential from a practical point of view as described above but it is also used to achieve that the unfolding construction preserves the firing behavior.

**Theorem 5.1.6 (Folding and Unfolding Constructions are Inverse)**
Let $N = (\Sigma, P, T, pre, cond, type)$ be an AHO-net scheme with $\Sigma = (S, OP, X)$, $c : type \in OP$, and $f_c : type \in X$.

1. Then the composition of the folding and unfolding constructions wrt. constant symbols leads to equivalent nets, i.e.

   $$U(F(N)) \cong N$$

   if the folding and unfolding constructions are wrt. the same constant symbol $c$ with corresponding variable $f_c$, the same transition is used by the folding and unfolding constructions, i.e. $t^{f_c} = t^c$ and the place generated by the folding construction is used by the unfolding construction, i.e. $p^{f_c} = p^c$.

2. Then the composition of the unfolding and folding constructions wrt. constant symbols leads to equivalent nets, i.e.

   $$F(U(N)) \cong N$$

   if the unfolding and folding constructions are wrt. the same constant symbol $c$ with corresponding variable $f_c$, the same transition is used by the unfolding and folding constructions, i.e. $t^c = t^{f_c}$ and the place used by the unfolding construction is generated by the folding construction, i.e. $p^c = p^{f_c}$.

**Proof**

1. In the AHO-net scheme $U(F(N))$ resp. the AHO-net scheme $N$ the signature is given by $\Sigma$ and the set of transitions is given by $T$. Furthermore we have

   $$\begin{aligned} U(F(P)) &= U(P \uplus \{p^c\}) \\ &= (P \uplus \{p^c\}) \setminus \{p^{f_c}\} \\ &= P. \end{aligned}$$

   We distinguish for $t \in T$ the following two cases. For $t \neq t^c$ the environment is preserved by the folding and unfolding constructions, i.e. we have

   $$\begin{aligned} &U(F(pre))(t) = pre(t), \\ &U(F(post))(t) = post(t), \text{ and} \\ &U(F(cond))(t) = cond(t). \end{aligned}$$

   Let $t = t^c$ and $pre(t^c) = \sum_{i=1}^{n}(term_i, p_i)$. Then we have

   $$\begin{aligned} & U(F(pre))(t^c) \\ = \; & U(\textstyle\sum_{i=1}^{n}(fold_c(term_i), p_i) \oplus (f_c, p^c)) \\ = \; & \textstyle\sum_{i=1}^{n}(unfold_{f_c}(fold_c(term_i)), p_i) \\ = \; & \textstyle\sum_{i=1}^{n}(term_i, p_i) && \text{(Fact 5.1.3)} \\ = \; & pre(t^c) \end{aligned}$$

   and analogously for the post domain. Furthermore, we have

   $$\begin{aligned} & U(F(cond))(t^c) \\ = \; & U(fold_c(cond(t^c))) \\ = \; & unfold_{f_c}(fold_c(cond(t^c))) \\ = \; & cond(t^c) && \text{(Fact 5.1.3).} \end{aligned}$$

   Finally, for all $p \in P$ we have $U(F(type))(p) = type(p)$ due to the definition of $F(type)(p) = type(p)$ and $U(type)(p) = type(p)$. Thus, we conclude $U(F(N)) \cong N$.

2. The proof of $F(U(N)) \cong N$ is more or less analog.

□

For our main theorem we need some preliminaries concerning the folding of terms and corresponding variable valuations. First we state, that the interpretation of terms is preserved by the fold-operator. Afterwards, we show, that the variable valuations in an AHO-net and the variable valuations in the folded AHO-net are in a correspondence, such that the definedness of terms is preserved.

**Lemma 5.1.7 (Folding Preserves Term Evaluation)**
Let $\Sigma = (S, OP, X)$ be a higher-order signature, $c : type \in OP$ a constant symbol and $f_c : type \in X$ a variable. Furthermore, let $A$ be a higher-order $\Sigma$-algebra such that $c_A \in A_{type}$ and $v : T_\Sigma(X) \longrightarrow A$ a variable valuation such that $v(f_c) = c_A$. Then we have for all $term \in T_\Sigma(X)$

$$v^\sharp(fold_c(term)) = v^\sharp(term).$$

**Proof:** By induction over the term structure of $T_\Sigma(X)$ we have:

**Induction Base:**

**1.** $v^\sharp(fold_c(\langle\rangle)) = v^\sharp(\langle\rangle)$ for $\langle\rangle \in T_{\Sigma,unit}(X)$

**2.** $v^\sharp(fold_c(x)) = v^\sharp(x)$ for all $x : type \in X$

**Induction Step:** Let $v^\sharp(fold_c(term_i)) = v^\sharp(term_i)$ for some terms $term_i \in T_\Sigma(X), i = 1, \ldots, n$.

**3a.** $v^\sharp(fold_c(op)) = v^\sharp(fold_c(c)) = v^\sharp(f_c) = c_A = v^\sharp(c) = v^\sharp(op)$

for $op : type \in OP$ and $op = c$

**3b.**

$$\begin{aligned} v^\sharp(fold_c(op(term))) &= v^\sharp(op(fold_c(term))) \\ &= op_A(v^\sharp(fold_c(term))) \\ &= op_A(v^\sharp(term)) \\ &= v^\sharp(op(term)) \end{aligned}$$

for all $op : type_1 \rightharpoonup type_2 \in OP, op \neq c$ and all $term \in T_{\Sigma,type_1}(X)$

**4.**

$$\begin{aligned} & v^\sharp(fold_c(\langle term_1, \ldots, term_n\rangle)) \\ = \; & v^\sharp(\langle fold_c(term_1), \ldots, fold_c(term_n)\rangle) \\ = \; & \langle v^\sharp(fold_c(term_1)), \ldots, v^\sharp(fold_c(term_n))\rangle \\ = \; & \langle v^\sharp(term_1), \ldots, v^\sharp(term_n)\rangle \\ = \; & v^\sharp(\langle term_1, \ldots, term_n\rangle) \end{aligned}$$

for all $term_1 \in T_{\Sigma,type_1}(X), \ldots, term_n \in T_{\Sigma,type_n}(X)$

**5.**

$$\begin{aligned} & v^\sharp(fold_c(pr_i^{(type_1 \star \ldots \star type_n)}(term))) \\ = \; & v^\sharp(pr_i^{(type_1 \star \ldots \star type_n)}(fold_c(term))) \\ = \; & pr_{i,A}^{(type_1 \star \ldots \star type_n)}(v^\sharp(fold_c(term))) \\ = \; & pr_{i,A}^{(type_1 \star \ldots \star type_n)}(v^\sharp(term)) \\ = \; & v^\sharp(pr_i^{(type_1 \star \ldots \star type_n)}(term)) \end{aligned}$$

for all $pr_i^{(type_1 \star \ldots \star type_n)} \in OP^{\rightarrow} \setminus OP$ and $term \in T_{\Sigma,(type_1 \star \ldots \star type_n)}(X)$. □

**Lemma 5.1.8 (Folding and Variable Valuation)**
Let $N = (\Sigma, P, T, pre, post, cond, type)$ be an AHO-net scheme with $\Sigma = (S, OP, X)$, $c : type \in OP$, $f_c : type \in X$ and $t^c \in T$. Let $(N, A)$ be an AHO-net wrt. $\Sigma$ such that $c_A \in A_{type}$. Moreover, let $F(N)$ be the folding of $N$ wrt. the constant symbol $c$ with corresponding variable $f_c$ and the transition $t^c$. Then for each variable valuation $v_1 : Var_{(N,A)}(t^c) \longrightarrow A$, there is a variable valuation $v_2 : Var_{(F(N),A)}(t^c) \longrightarrow A$ defined by

$$v_2(x) = \begin{cases} c_A & \text{if } x = f_c \\ v_1(x) & \text{else} \end{cases}$$

and for each variable valuation $v_2 : Var_{(F(N),A)}(t^c) \longrightarrow A$ there is a variable valuation $v_1 : Var_{(N,A)}(t^c) \longrightarrow A$ defined by

$$v_1 = v_{2|Var_{(N,A)}(t^c)}.$$

Moreover, we have for all $term \in T_\Sigma(X)$ with $f_c \notin Var(term)$

$$term \in dom(v_1^\sharp) \Longleftrightarrow term \in dom(v_2^\sharp).$$

**Proof:** Due to the assumption in Fact 5.1.5 we have $f_c \notin Var_{(N,A)}(t^c)$. Let $v_1 : Var_{(N,A)}(t^c) \longrightarrow A$ be a variable valuation. Then $v_2$ given by the extension of $v_1$ to the variable $f_c$ is well-defined because $v_1$ is well-defined, $Var_{(F(N),A)}(t^c) = Var_{(N,A)}(t^c) \cup \{f_c\}$, and for $f_c : type \in X$ we have $v_2(f_c) = c_A \in A_{type}$.

Given $term \in dom(v_1^\sharp)$ and $f_c \notin Var(term)$, then due to the definition of $v_2$ we have $v_2^\sharp(term) = v_1^\sharp(term)$ and, thus, $term \in dom(v_2^\sharp)$.

Let $v_2 : Var_{(F(N),A)}(t^c) \longrightarrow A$ be a variable valuation. Then $v_1$ given by the restriction of $v_2$ to the variables $Var_{(N,A)}(t^c)$ is well-defined because $v_2$ is well-defined and $Var_{(N,A)}(t^c) = Var_{(F(N),A)}(t^c) \setminus \{f_c\}$.

Given $term \in dom(v_2^\sharp)$ and $f_c \notin Var(term)$, then due to the definition of $v_1$ we have $v_1^\sharp(term) = v_2^\sharp(term)$ and, thus, $term \in dom(v_1^\sharp)$. □

Our second theorem is divided into two main results. We provide that not only the folding construction wrt. constant symbols but also the unfolding construction wrt. constant symbols preserves the operational behavior. Within this theorem the AHO-nets are in some sense parametrized by a fixed marking of the contextual places, which are used in these constructions. But to simplify our notation, we give these parameters only implicitly. In detail we show that the (un-)folded AHO-net has exactly the same sets of markings, consistent transitions valuations, enabled transitions, firing steps, and reachable markings as the AHO-net, and thus the two nets are equivalent wrt. their firing behavior.

**Theorem 5.1.9 (Folding and Unfolding wrt. Constant Symbols)**
Let $N = (\Sigma, P, T, pre, post, cond, type)$ be an AHO-net scheme with $\Sigma = (S, OP, X)$, $c : type \in OP$, and $f_c : type \in X$. Let $A$ a higher-order partial $\Sigma$-algebra such that $c_A \in A_{type}$.

1. Let $F(N)$ be the folding of $N$ wrt. the constant symbol $c$ with corresponding variable $f_c$ and the transition $t^c \in T$. Let $p^c$ be the contextual place generated by the folding construction. Then the AHO-net $(N, A)$ and the AHO-net $(F(N), A)$ are equivalent wrt. their firing behavior, if $M_{(F(N),A)|p^c} = (c_A, p^c)$. More precisely,

   (a) the markings are in a bijective correspondence:
   $$M_{(N,A)} \in CP^{\oplus}_{(N,A)} \Longleftrightarrow M_{(F(N),A)} \in CP^{\oplus}_{(F(N),A)}$$

(b) the sets of consistent transition valuations are in a bijective correspondence:

$(t, v_1) \in CT_{(N,A)} \Longleftrightarrow (t, v_2) \in CT_{(F(N),A)}$

(c) the sets of enabled transition are equivalent:

$M_{(N,A)}[(t, v_1)\rangle \Longleftrightarrow M_{(F(N),A)}[(t, v_2)\rangle$

(d) the sets of firing steps are equivalent:

$M_{(N,A)}[(t, v_1)\rangle M'_{(N,A)} \Longleftrightarrow M_{(F(N),A)}[(t, v_2)\rangle M'_{(F(N),A)}$

(e) and the sets of reachable markings are equivalent:

$M'_{(N,A)} \in [M_{(N,A)}\rangle \Longleftrightarrow M'_{(F(N),A)} \in [M_{(F(N),A)}\rangle$.

2. Let $U(N)$ be the unfolding of $N$ wrt. the constant symbol $c$ with corresponding variable $f_c$, the transition $t^{f_c} \in T$ and the contextual place $p^{f_c} \in P$. Then the AHO-net $(N, A)$ and the AHO-net $(U(N), A)$ are equivalent wrt. their firing behavior, if $M_{(N,A)|p^{f_c}} = (c_A, p^{f_c})$. More precisely,

(a) the sets of markings are in a bijective correspondence:

$M_{(N,A)} \in CP^{\oplus}_{(N,A)} \Longleftrightarrow M_{(U(N),A)} \in CP^{\oplus}_{(U(N),A)}$

(b) the sets of consistent transition valuations are in a bijective correspondence:

$(t, v_1) \in CT_{(N,A)} \Longleftrightarrow (t, v_2) \in CT_{(U(N),A)}$

(c) the sets of enabled transition are equivalent:

$M_{(N,A)}[(t, v_1)\rangle \Longleftrightarrow M_{(U(N),A)}[(t, v_2)\rangle$

(d) the sets of firing steps are equivalent:

$M_{(N,A)}[(t, v_1)\rangle M'_{(N,A)} \Longleftrightarrow M_{(U(N),A)}[(t, v_2)\rangle M'_{(U(N),A)}$

(e) and the set of reachable markings are equivalent:

$M'_{(N,A)} \in [M_{(N,A)}\rangle \Longleftrightarrow M'_{(U(N),A)} \in [M_{(U(N),A)}\rangle$.

**Proof:** The proof is a consequence of the definition of the folding and unfolding constructions and the definition of the firing behavior of AHO-nets.

**1. (N,A) and (F(N),A) are equivalent wrt. their firing behavior.**
In the following we assume that for a marking $M_{(F(N),A)}$ of the AHO-net $(F(N), A)$ we have $M_{(F(N),A)|p^c} = (c_A, p^c)$. $M_{(F(N),A)|p^c} \in CP^{\oplus}_{(F(N),A)}$, because $F(P) = P \uplus \{p^c\}$, $c_A \in A_{type}$, $F(type)(p^c) = type$ and $p^c \in F(P)$. Note that due to the definition of the folding construction we have for $t \in T$ and $t \neq t^c$:

$$F(pre)(t) = pre(t), F(post)(t) = post(t) \text{ and } F(cond)(t) = cond(t). \quad (5.1)$$

For $t \in T$ and $t = t^c$ we have:

$$F(cond)(t^c) = fold_c(cond)(t^c). \quad (5.2)$$

Let $pre(t^c) = \sum_{i=1}^{n}(term_i, p_i)$ and $post(t^c) = \sum_{j=1}^{m}(term_j, p_j)$, then we have:

$$F(pre)(t^c) = \sum_{i=1}^{n}(fold_c(term_i), p_i) \oplus (f_c, p^c) \quad (5.3)$$

and

$$F(post)(t^c) = \sum_{j=1}^{m}(fold_c(term_j), p_j) \oplus (f_c, p^c). \quad (5.4)$$

**(a)** $\Rightarrow$ Let $M_{(N,A)} \in CP^{\oplus}_{(N,A)}$ be a marking of the AHO-net $(N, A)$. Then the marking of the AHO-net $(F(N), A)$ is defined by

$$M_{(F(N),A)} = M_{(N,A)} \oplus (c_A, p^c).$$

$M_{(F(N),A)}$ is well-defined because $F(P) = P \uplus \{p^c\}$.

$\Leftarrow$ Let $M_{(F(N),A)} \in CP^{\oplus}_{(F(N),A)}$ be a marking of the AHO-net $(F(N), A)$. Then the marking of the AHO-net $(N, A)$ is defined by

$$M_{(N,A)} = M_{(F(N),A)|F(P)\setminus\{p^c\}}.$$

$M_{(N,A)}$ is well-defined because $P = F(P) \setminus \{p^c\}$.

**(b)** It remains to show for all $t \in T$:

$$\begin{aligned}
& cond(t) \in dom(v_1^\sharp) \text{ and} \\
& \forall (term, p) \in pre(t) \oplus post(t) : term \in dom(v_1^\sharp) \\
\iff\ & F(cond)(t) \in dom(v_2^\sharp) \text{ and} \\
& \forall (term', p') \in F(pre)(t) \oplus F(post)(t) : term' \in dom(v_2^\sharp).
\end{aligned}$$

We distinguish for $t \in T$ the following two cases.

**Case 1:** For $t \neq t^c$ it follows immediately due to (5.1) that

$$(t, v) \in CT_{(N,A)} \iff (t, v) \in CT_{(F(N),A)}.$$

**Case 2:** Let $t = t^c$ and $v_1 : Var_{(N,A)}(t^c) \longrightarrow A$ be a variable valuation. Due to the folding construction $Var_{(N,A)}(t^c)$ and $Var_{(F(N),A)}(t^c)$ differ at least in one variable. So we can define a variable valuation $v_2 : Var_{(F(N),A)}(t^c) \longrightarrow A$ as an extension of $v_1$ according to the marking of the contextual place $p^C$. Vice versa we assume a variable valuation $v_2 : Var_{(F(N),A)}(t^c) \longrightarrow A$. Then $v_1$ is given by the restriction of $v_2$ to the variables $Var_{(N,A)}(t^c)$ (see Lemma 5.1.8).

$$\begin{aligned}
& cond(t^c) \in dom(v_1^\sharp) && \\
\iff\ & cond(t^c) \in dom(v_2^\sharp) && \text{(Lemma 5.1.8)} \\
\iff\ & fold_c(cond(t^c)) \in dom(v_2^\sharp) && \text{(Lemma 5.1.7)} \\
\iff\ & F(cond)(t^c) \in dom(v_2^\sharp) && \text{(by (5.2))}.
\end{aligned}$$

We have for all $(term, p) \in pre(t^c) \oplus post(t^c)$ with $(term, p) \neq (f_c, p^c)$ the following.

$$\begin{aligned}
& \forall (term, p) \in pre(t^c) \oplus post(t^c) : term \in dom(v_1^\sharp) \\
\iff\ & \forall (term, p) \in pre(t^c) \oplus post(t^c) : term \in dom(v_2^\sharp) \\
& && \text{(Lemma 5.1.8)} \\
\iff\ & \forall (term, p) \in pre(t^c) \oplus post(t^c) : fold_c(term) \in dom(v_2^\sharp) \\
& && \text{(Lemma 5.1.7)} \\
\iff\ & \forall (fold_c(term), p) \in F(pre)(t^c) \oplus F(post)(t^c) : \\
& \qquad fold_c(term) \in dom(v_2^\sharp) && \text{(by (5.3) and (5.4))}.
\end{aligned}$$

Moreover, $f_c \in dom(v_2^\sharp)$ due to the definition of $v_2$ in Lemma 5.1.8 and we have:

$$(f_c, p^c) \in F(pre)(t^c) \oplus F(post)(t^c) : f_c \in dom(v_2^\sharp).$$

Thus

$$\begin{aligned} & \forall (term, p) \in pre(t^c) \oplus post(t^c) : term \in dom(v_1^\sharp) \\ \iff \quad & \forall (term', p') \in F(pre)(t^c) \oplus F(post)(t^c) : term' \in dom(v_2^\sharp). \end{aligned}$$

**(c)** It remains to show for all $t \in T$:

$$\widehat{v_1}(pre(t)) \leq M_{(N,A)} \iff \widehat{v_2}(F(pre)(t)) \leq M_{(F(N),A)}.$$

We distinguish for $t \in T$ the following two cases:

**Case 1:** For $t \neq t^c$ we have $p^c \notin {}^\bullet t$. Thus, it follows by (5.1) and Proofs of **(a)** and **(b)**:

$$\begin{aligned} & \widehat{v}(pre(t)) \leq M_{(N,A)} \\ \iff \quad & \widehat{v}(F(pre)(t)) \leq M_{(N,A)} \oplus (c_A, p^c) \\ \iff \quad & \widehat{v}(F(pre)(t)) \leq M_{(F(N),A)}. \end{aligned}$$

**Case 2:** Let $t = t^c$ and $pre(t^c) = \sum_{i=1}^n (term_i, p_i)$.

$$\begin{aligned} & \widehat{v_1}(pre(t^c)) \leq M_{(N,A)} \\ \iff \quad & \textstyle\sum_{i=1}^n (v_1^\sharp(term_i), p_i) \leq M_{(N,A)} \\ \iff \quad & \textstyle\sum_{i=1}^n (v_2^\sharp(term_i), p_i)) \leq M_{(N,A)} \\ & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(Lemma 5.1.8)} \\ \iff \quad & \textstyle\sum_{i=1}^n (v_2^\sharp(term_i), p_i)) \oplus (c_A, p^c) \leq M_{(N,A)} \oplus (c_A, p^c) \\ \iff \quad & \textstyle\sum_{i=1}^n (v_2^\sharp(term_i), p_i)) \oplus (v_2^\sharp(f_c), p^c)) \leq M_{(N,A)} \oplus (c_A, p^c) \\ & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(Lemma 5.1.8)} \\ \iff \quad & \textstyle\sum_{i=1}^n (v_2^\sharp(fold_c(term_i)), p_i)) \oplus (v_2^\sharp(f_c), p^c)) \\ & \qquad\qquad\qquad\qquad\qquad\qquad \leq M_{(N,A)} \oplus (c_A, p^c) \\ & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(Lemma 5.1.7)} \\ \iff \quad & \widehat{v_2}(\textstyle\sum_{i=1}^n (fold_c(term_i), p_i) \oplus (f_c, p^c)) \leq M_{(N,A)} \oplus (c_A, p^c) \\ \iff \quad & \widehat{v_2}(F(pre)(t^c)) \leq M_{(N,A)} \oplus (c_A, p^c) \qquad \text{(by (5.3))} \\ \iff \quad & \widehat{v_2}(F(pre)(t^c)) \leq M_{(F(N),A)} \qquad \text{(Proof of \textbf{(a)}).} \end{aligned}$$

**(d)** Given $M_{(N,A)}[(t, v_1)\rangle$ and $M_{(F(N),A)}[(t, v_2)\rangle$ such that

$$M_{(N,A)}[(t, v_1)\rangle \iff M_{(F(N),A)}[(t, v_2)\rangle,$$

due to the definition of follower markings $M'_{(N,A)}$ we have:

$$\begin{aligned} & M'_{(N,A)} = M_{(N,A)} \ominus \widehat{v_1}(pre(t)) \oplus \widehat{v_1}(post(t)) \\ \iff \quad & M'_{(N,A)} \oplus (c_A, p^c) = M_{(N,A)} \oplus (c_A, p^c) \\ & \qquad\qquad\qquad\qquad \ominus \widehat{v_1}(pre(t)) \oplus \widehat{v_1}(post(t)) \\ \iff \quad & M'_{(F(N),A)} = M_{(F(N),A)} \ominus \widehat{v_1}(pre(t)) \oplus \widehat{v_1}(post(t)). \end{aligned}$$

where the last equivalence follows due to Proof of **(a)**. Moreover we have for the follower marking $M'_{(F(N),A)}$:

$$M'_{(F(N),A)} = M_{(F(N),A)} \ominus \widehat{v_2}(F(pre)(t)) \oplus \widehat{v_2}(F(post)(t)).$$

It remains to show that for all $t \in T$:

$$\widehat{v_1}(post(t)) \ominus \widehat{v_1}(pre(t)) = \widehat{v_2}(F(post)(t)) \ominus \widehat{v_2}(F(pre)(t)).$$

We distinguish for $t \in T$ the following two cases.

**Case 1:** For $t \neq t^c$ the equation follows by (5.1) and $v_1 = v_2$.
**Case 2:** Let $t = t^c$ with

$$pre(t^c) = \sum_{i=1}^{n}(term_i, p_i) \text{ and } post(t^c) = \sum_{j=1}^{m}(term_j, p_j).$$

$$\begin{aligned}
& \widehat{v_1}(post(t^c)) \ominus \widehat{v_1}(pre(t^c)) \\
= \;& \textstyle\sum_{j=1}^{m}(v_1^\sharp(term_j), p_j) \ominus \sum_{i=1}^{n}(v_1^\sharp(term_i), p_i) \\
= \;& \textstyle\sum_{j=1}^{m}(v_2^\sharp(term_j), p_j) \ominus \sum_{i=1}^{n}(v_2^\sharp(term_i), p_i) \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(Lemma 5.1.8)} \\
= \;& \textstyle\sum_{j=1}^{m}(v_2^\sharp(fold_c(term_j)), p_j) \ominus \sum_{i=1}^{n}(v_2^\sharp(fold_c(term_i)), p_i) \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(Lemma 5.1.7)} \\
= \;& \textstyle\sum_{j=1}^{m}(v_2^\sharp(fold_c(term_j)), p_j)) \oplus (v_2^\sharp(f_c), p^c) \\
& \qquad \ominus \sum_{i=1}^{n}(v_2^\sharp(fold_c(term_i)), p_i) \ominus (v_2^\sharp(f_c), p^c) \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(Lemma 5.1.8)} \\
= \;& \widehat{v_2}(\textstyle\sum_{j=1}^{m}(fold_c(term_j), p_j) \oplus (f_c, p^c)) \\
& \qquad \ominus \widehat{v_2}(\textstyle\sum_{i=1}^{n}(fold_c(term_i), p_i) \oplus (f_c, p^c)) \\
= \;& \widehat{v_2}(F(post)(t^c)) \ominus \widehat{v_2}(F(pre)(t^c)) \qquad \text{(by (5.3) and (5.4))}.
\end{aligned}$$

**(e)** We show by induction over the length of occurrence sequences that the set of reachable markings are equivalent. For $n = 0$ it follows immediately due to the Proof of **(a)** that

$$M_{(N,A)} \in [M_{(N,A)}\rangle \iff M_{(F(N),A)} \in [M_{(F(N),A)}\rangle.$$

For $n = 1$ there are firing steps

$$M_{(N,A)}[(t, v_1)\rangle M'_{(N,A)} \text{ and } M_{(F(N),A)}[(t, v_2)\rangle M'_{(F(N),A)}$$

due to Proof of **(d)**. Hence,

$$M'_{(N,A)} \in [M_{(N,A)}\rangle \iff M'_{(F(N),A)} \in [M_{(F(N),A)}\rangle.$$

For occurrence sequences of the length $n+1$ it follows due to the induction base and the induction hypothesis that for the end markings we have:

$$M_{(N,A),n+2} \in [M_{(N,A)}\rangle \iff M_{(F(N),A),n+2} \in [M_{(F(N),A)}\rangle.$$

**2. (N,A) and (U(N),A) are equivalent wrt. their firing behavior.** Let $U(N)$ be the unfolding of $N$ wrt. the constant symbol $c$ with corresponding variable $f_c$, the transition $t^{f_c} \in T$ and the contextual place $p^{f_c} \in P$. Moreover, let $F$ be a folding construction, such that $F(U(N)) \cong N$ (see Thm. 5.1.6). We define $U(N) = N'$. Then we have due to **1.(a)**

$$\begin{aligned}
& (M_{(N',A)} \in CP^{\oplus}_{(N',A)} \iff M_{(F(N'),A)} \in CP^{\oplus}_{(F(N'),A)}) \\
\implies \;& (M_{(U(N),A)} \in CP^{\oplus}_{(U(N),A)} \iff M_{(F(U(N)),A)} \in CP^{\oplus}_{(F(U(N)),A)}) \\
\implies \;& (M_{(U(N),A)} \in CP^{\oplus}_{(U(N),A)} \iff M_{(N,A)} \in CP^{\oplus}_{(N,A)})
\end{aligned}$$

and analogously for **2.(b) - (e)**.

□

## 5.2 Folding and Unfolding wrt. Product Types

In this section we introduce the concept of the folding and unfolding construction wrt. product types. The folding of AHO-net schemes wrt. product types is defined for a distinguished set $P^F$ of at least two places, called uniform places, so that all places $p \in P^F$ have the same environment, i.e. the same set of transitions in the pre- and post domains and an unary arc inscription for arcs between a place $p \in P^F$ and transitions in the pre- and post domain of $p$. In terms of our example in Section 3.3 the set $\{p_1, p_2\}$ of the AHO-net "Computation I" (see Fig. 3.1) is a set of uniform places because the pre- and post domain of $p_1$ resp. $p_2$ is given by the transitions *compute add* and *compute sub* and there are unary arc inscription $x$ resp. $y$.

**Definition 5.2.1 (Set of Uniform Places)**
Given an AHO-net scheme $N = (\Sigma, P, T, pre, post, cond, type)$. Then a set $P^F$ with $P^F \subseteq P$ and $P^F = \{p_1, \ldots, p_n\}, n \geq 2$, where $p_1, \ldots, p_n$ pairwise distinct, is a set of uniform places if there is a subset $T^F \subseteq T$, called the pre- and post domain of $P^F$, such that for all $p \in P^F$:

$$\bullet p = p\bullet = T^F$$

and for all $t \in T^F$ there exists a $term \in T_\Sigma(X)$ such that

$$pre(t)_{|p} = post(t)_{|p} = term.$$

For $type(p_i) = type_i$, and $i \in \{1, \ldots, n\}$ we also denote $P^F$ by the corresponding list $\langle p_1 : type_1, \ldots, p_n : type_n \rangle$.

**Remark 5.2.2 (Pre- and Post Domain of Set of Uniform places)**
For $t \in T^F$ and $P^F = \langle p_1 : type_1, \ldots, p_n : type_n \rangle$ the restriction of the pre- and post domains to $P^F$ is given by

$$pre(t)_{|P^F} = post(t)_{|P^F} = \sum_{i=1}^{n} (term_i, p_i) \text{ in normal form.}$$

Next we define the folding construction of an AHO-net scheme wrt. product types. The folding construction is applied to a specific set of uniform places. The data type part, the set of transitions, and the firing conditions are remain unchanged. The set of places of the folded AHO-net is given by deleting the set of uniform places but adding a new place with an appropriate product type. Furthermore, the arc inscriptions of the transitions in the environment of the set of uniform places are obtained by using the tuple formation. A general description of the folding construction wrt. product types is depicted in Fig. 5.3.

**Fact 5.2.3 (Folding of AHO-net schemes w.r.t. Product Types)**
Let $N = (\Sigma, P, T, pre, post, cond, type)$ be an AHO-net scheme and let $P^F \subseteq P$ be a set of uniform places with corresponding list $\langle p_1 : type_1, \ldots, p_n : type_n \rangle, n \geq 2$, and $T^F \subseteq T$ the pre- and post domain of $P^F$. Then the folding of $N$ wrt. products types and $P^F$ is an AHO-net scheme

$$F(N) = (F(\Sigma), F(P), F(T), F(pre), F(post), F(cond), F(type))$$

defined by

- $F(\Sigma) = \Sigma$,
- $F(P) = (P \setminus P^F) \uplus \{p^F\}$ where $p^F$ is the place corresponding to the folding of the set of uniform places $P^F$,

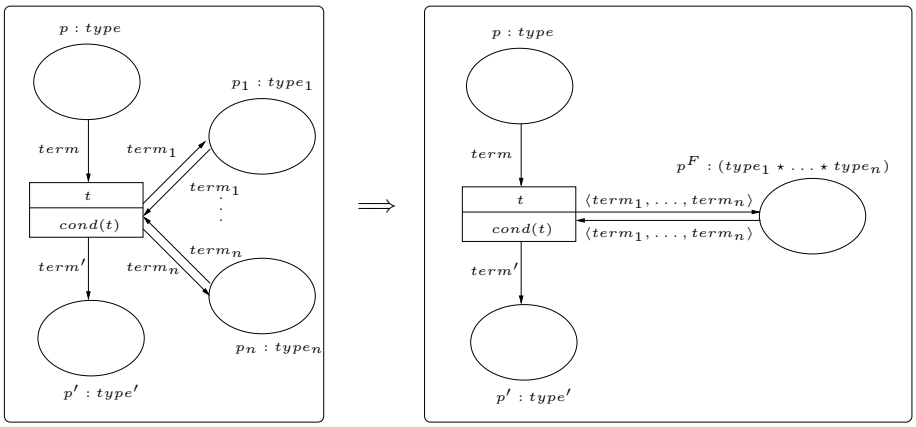



Figure 5.3: Schemata of folding construction wrt. product types

- $F(T) = T$,
- $F(pre) : T \longrightarrow (T_\Sigma(X) \otimes F(P))^\oplus$ with

$$F(pre)(t) = \begin{cases} pre(t) \ominus pre(t)_{|P^F} \oplus (\langle term_1, \ldots, term_n \rangle, p^F) \\ \qquad\qquad \text{if } t \in T^F \text{ and } pre(t)_{|P^F} = \sum_{i=1}^{n} (term_i, p_i) \\ pre(t) \qquad \text{else} \end{cases}$$

and $F(post) : T \longrightarrow (T_\Sigma(X) \otimes F(P))^\oplus$ with

$$F(post)(t) = \begin{cases} post(t) \ominus post(t)_{|P^F} \oplus (\langle term_1, \ldots, term_n \rangle, p^F) \\ \qquad\qquad \text{if } t \in T^F \text{ and } post(t)_{|P^F} = \sum_{i=1}^{n} (term_i, p_i) \\ post(t) \qquad \text{else} \end{cases}$$

- $F(cond) = cond$, and
- $F(type) : F(P) \longrightarrow S^\rightarrow$ with

$$F(type)(p) = \begin{cases} type_1 \star \ldots \star type_n & \text{for } p = p^F \text{ and} \\ & P^F = \langle p_1 : type_1, \ldots, p_n : type_n \rangle \\ type(p) & \text{else} \end{cases}$$

**Proof (Sketch):** We have to show that $F(N)$ is an AHO-net scheme as defined in Def. 3.2.1. In the following we restrict our proof sketch to the set of transitions $t \in T^F$ and the place $p^F$ generated by the folding construction. $p^F \in F(P)$ and $term_1, \ldots, term_n \in T_\Sigma(X)$ implies $(\langle term_1, \ldots, term_n \rangle, p^F) \in (T_\Sigma(X) \otimes F(P))^\oplus$ (see Remark 3.1.15). Thus, we have $F(pre)(t) \in (T_\Sigma(X) \otimes F(P))^\oplus$; analogously for the post domain. Moreover, $F(type)(p^F) \in S^\rightarrow$ because $type_1, \ldots, type_n \in S^\rightarrow$ implies $type_1 \star \ldots \star type_n \in S^\rightarrow$ (see Remark 3.1.5). □

An example of the folding construction wrt. product types can be found in Section 3.3, where we apply the folding construction wrt. the set of uniform places $P^F = \{p_1, p_2\}$ to the AHO-net "Computation I" in Fig. 3.1. Note that $type(p_1) = type(p_2) = Nat$. In more detail, we replace the set of uniform places $P^F$ by the

new place $p^F$ with assigned product type $Nat \star Nat$. Moreover, we get the arc inscription $\langle x, y \rangle$ corresponding to the unary arc inscriptions $x$ resp. $y$. As a result we obtain the AHO-net "Computation II" in Fig. 3.2.

Next we define the notion of uniform places, which is used in the unfolding construction. A place $p^U \in P$ is called a uniform place, if there are the same transitions with unary arc inscriptions in its environment. An example of an uniform place is given by the place $p^F$ in the AHO-net "Computation II" (see Fig. 3.2).

**Definition 5.2.4 (Uniform Place)**
Given an AHO-net scheme $N = (\Sigma, P, T, pre, post, cond, type)$, then a place $p^U \in P$ with assigned product type $(type_1 \star \ldots \star type_n) \in S^{\rightarrow}, n \geq 2$, is a uniform place iff there is a set $T^{p^U} \subseteq T$, called the pre- and post domain of $p^U$ such that

$$p^U\bullet = \bullet p^U = T^{p^U}$$

and for all $t \in T^{p^U}$ there exists a term $\langle term_1, \ldots, term_n \rangle \in T_\Sigma(X)$, such that

$$pre(t)_{|p^U} = post(t)_{|p^U} = \langle term_1, \ldots, term_n \rangle.$$

**Remark 5.2.5 (Pre- and Post Domain of Uniform Places)**
Here we allow that $type(p^U)$ is a nested product type (see Remark 3.1.5) and $\langle term_1, \ldots, term_n \rangle$ is a nested product term (see Remark 3.1.15).

In the following we define the unfolding construction of an AHO-net scheme and a specific uniform place with assigned product type. This construction has no effect on the data type part, the set of transitions, and the firing conditions, while the set of places of the unfolded AHO-net is given by deleting the uniform place and adding a specific set of places. The number of added places corresponds to the number of subtypes, from which the product type is constructed. Analogously the type assigned to these places corresponds in some sense to the product type of the uniform place. Furthermore, the arc inscriptions of the transitions in the environment of the uniform place are unfolded, i.e. the particular arc inscriptions are obtained by splitting the tuple into several subterms. If we exchange the left-hand side and the right-hand side of Fig. 5.3, the resulting graphical description illustrates the unfolding construction wrt. product types.

**Fact 5.2.6 (Unfolding of AHO-net schemes w.r.t. Product Types)**
Let $N = (\Sigma, P, T, pre, post, cond, type)$ be an AHO-net scheme and let $p^U \in P$ be a uniform place with $type(p^U) = (type_1 \star \ldots \star type_n), n \geq 2$. Then the unfolding of $N$ wrt. products types is an AHO-net scheme

$$U(N) = (U(\Sigma), U(P), U(T), U(pre), U(post), U(cond), U(type))$$

defined by

- $U(\Sigma) = \Sigma$,
- $U(P) = (P \setminus \{p^U\}) \uplus P^U$ where $P^U$ is the set of places corresponding to the unfolding of the uniform place $p^U$ with $P^U = \{p_1, \ldots, p_n\}, n \geq 2, p_1, \ldots, p_n$ pairwise distinct and corresponding list $\langle p_1 : type_1, \ldots, p_n : type_n \rangle$,
- $U(T) = T$,
- $U(pre) : T \longrightarrow (T_\Sigma(X) \otimes U(P))^\oplus$ with

$$U(pre)(t) = \begin{cases} pre(t) \ominus pre(t)_{|p^U} \oplus \sum_{i=1}^{n} (term_i, p_i), \\ \qquad \text{if } t \in \bullet p^U \text{ with } pre(t)_{|p^U} = \langle term_1, \ldots, term_n \rangle \\ \qquad \text{and } P^U = \langle p_1 : type_1, \ldots, p_n : type_n \rangle \\ pre(t) \quad \text{else} \end{cases}$$

and $U(post) : T \longrightarrow (T_\Sigma(X) \otimes U(P))^\oplus$ with

$$U(post)(t) = \begin{cases} post(t) \ominus post(t)_{|p^U} \oplus \sum_{i=1}^{n}(term_i, p_i), \\ \qquad \text{if } t \in {}^\bullet p^U \text{ with } post(t)_{|p^U} = \langle term_1, \ldots, term_n \rangle \\ \qquad \text{and } P^U = \langle p_1 : type_1, \ldots, p_n : type_n \rangle \\ post(t) \quad \text{else} \end{cases}$$

- $U(cond) = cond$, and
- $U(type) : U(P) \longrightarrow S^\rightarrow$ with

$$U(type)(p) = \begin{cases} type_i & \text{if } p = p_i \in P^U, i \in \{1, \ldots, n\}, \\ & \text{and } P^U = \langle p_1 : type_1, \ldots, p_n : type_n \rangle \\ type(p) & \text{else} \end{cases}$$

**Proof (Sketch):** We have to show that $U(N)$ is an AHO-net scheme as defined in Def. 3.2.1. In the following we restrict our proof sketch to the set of transition $t \in {}^\bullet p^U$ and the set of places $P^U \subseteq U(P)$ with corresponding list $\langle p_1 : type_1, \ldots, p_n : type_n \rangle$ generated by the unfolding construction. We have $U(pre)(t) \in (T_\Sigma(X) \otimes U(P))^\oplus$ because $pre(t)_{|p^U} = \langle term_1, \ldots, term_n \rangle$ implies $(term_1, p_1), \ldots, (term_n, p_n) \in (T_\Sigma(X) \otimes U(P))^\oplus$ (see Remark 5.2.5); analogously for the post domain. For all $p_i \in P^U$ we have $U(type)(p_i) \in S^\rightarrow$ because $type_1 \star \ldots \star type_n \in S^\rightarrow$ implies $type_1, \ldots, type_n \in S^\rightarrow$ (see Remark 5.2.5). □

Next we show that the composition of the folding and unfolding construction wrt. product types leads to equivalent nets.

**Theorem 5.2.7 (Folding and Unfolding Constructions are Inverse)**
Let $N$ be a AHO-net scheme.

1. Then the composition of the folding and unfolding construction wrt. product types leads to equivalent nets, i.e.

   $$U(F(N)) \cong N$$

   if the uniform place generated by the folding construction is used by the unfolding construction, i.e. $p^F = p^U$ and the set of uniform places used by the folding construction is generated by the unfolding construction, i.e. $P^F = P^U$.

2. Then the composition of the unfolding and folding construction wrt. product types leads to equivalent nets, i.e.

   $$F(U(N)) \cong N$$

   if the uniform place used by the unfolding construction is generated by the folding construction, i.e. $p^U = p^F$, and the set of uniform places generated by the unfolding construction is used by the folding construction, i.e. $P^U = P^F$.

**Proof:**

1. Let $N = (\Sigma, P, T, pre, cond, type)$ be an AHO-net scheme. Let the folding construction wrt. the set of uniform places $P^F \subseteq P$ with corresponding list $\langle p_1 : type_1, \ldots, p_n : type_n \rangle, n \geq 2$, and the unfolding construction wrt. the uniform place $p^U$ with $type(p^U) = type_1 \star \ldots \star type_n$.

Then in both the AHO-net scheme $U(F(N))$ and the AHO-net scheme $N$ the signature is given by $\Sigma$, the set of transitions is given by $T$ and the firing condition function is given by *cond*. Furthermore we have

$$\begin{aligned} U(F(P)) &= U((P \setminus P^F) \uplus \{p^F\}) \\ &= (((P \setminus P^F) \uplus \{p^F\}) \setminus \{p^U\}) \uplus P^U \\ &= P. \end{aligned}$$

We distinguish for $t \in T$ the following cases. For $t \neq T^F$ the pre-and post domains are preserved by the folding and unfolding construction, i.e. we have

$$\begin{aligned} U(F(pre))(t) = pre(t) \text{ and} \\ U(F(post))(t) = post(t). \end{aligned}$$

Let $t \in T^F$ and $pre(t)_{|P^F} = \sum_{i=1}^{n}(term_i, p_i)$. Then we have

$$\begin{aligned} & U(F(pre))(t) \\ = \;& U(pre(t) \ominus pre(t)_{|P^F} \oplus (\langle term_1, \ldots, term_n \rangle, p^F)) \\ = \;& pre(t) \ominus pre(t)_{|P^F} \oplus (\langle term_1, \ldots, term_n \rangle, p^F) \\ & \qquad \ominus pre(t)_{|p^U} \oplus \textstyle\sum_{i=1}^{n}(term_i, p_i) \\ = \;& pre(t) \end{aligned}$$

because $pre(t)_{|p^U} = (\langle term_1, \ldots, term_n \rangle, p^U)$. Finally, for all $p \in P$ we have $type(p) = U(F(type))(p)$ due to the definition of $F(type)(p) = type(p)$ and $U(type)(p) = type(p)$. Thus, we conclude $U(F(N)) \cong N$.

2. The proof $F(U(N)) \cong N$ is more or less analog.

□

To show the equivalence concerning the firing behavior, we need the notion of regular and cartesian markings. In our construction defined above we use the well-known operation of Cartesian products. Our next goal is to apply this operation to markings of AHO-nets. For this reason markings have to be of a special structure, so that they are not empty and rather sets than a multisets.

**Definition 5.2.8 (Regular Marking)**
Let $(N, A)$ be a AHO-net with $N = (\Sigma, P, T, pre, post, cond, type)$ and $M \in CP^{\oplus}$ be a marking of $(N, A)$. Then for $p \in P$ the marking $M_{|p}$ is regular if there is a finite and non empty subset $A_1 \subseteq A_{type(p)}$ such that $M_{|p} = \sum_{a \in A_1} a$, i.e. $M_{|p}$ is of the form $a_1 \oplus \ldots \oplus a_n, n \geq 1$, and $a_1, \ldots, a_n$ are pairwise distinct. Let $P^F \subseteq P$. Then $M_{|P^F}$ is regular if for all $p \in P^F$ $M_{|p}$ is regular.

**Definition 5.2.9 (Cartesian Marking)**
Let $(N, A)$ be a AHO-net with $N = (\Sigma, P, T, pre, post, cond, type)$ and $M \in CP^{\oplus}$ be a marking of $(N, A)$. Then for $p \in P$ with $type(p) = (type_1 \star \ldots \star type_n), n \geq 2$, the marking $M_{|p}$ is cartesian if there is a finite and non empty subset

$$A_1 \times \ldots \times A_n \subseteq A_{type_1} \times \ldots \times A_{type_n}$$

such that

$$M_{|p} = \sum_{(a_1, \ldots, a_n) \in (A_1 \times \ldots \times A_n)} (a_1, \ldots, a_n).$$

Note that a cartesian marking is regular.

The marking in the AHO-net "Computation I" in Fig. 3.1 is regular, while the marking of the place $p^F$ in the AHO-net "Computation II" in Fig. 3.2 is cartesian.

**Lemma 5.2.10 (Corresponding Cartesian Marking)**
Let $(N, A)$ be a AHO-net with $N = (\Sigma, P, T, pre, post, cond, type)$. Let $F(N)$ be the folding of $N$ wrt. product types and the set of uniform places $P^F \subseteq P$ with $P^F = \{p_1, \ldots, p_n\}$. Let $M_{(N,A)}(P^F)$ be an arbitrary but fixed marking of the set of uniform places such that $M_{(N,A)|P^F}$ is regular, i.e. for all $p_i \in P^F$ and $i \in \{1, \ldots, n\}$ there is a finite and non empty subsets $A_i \subseteq A_{type(p_i)}$ such that $M_{(N,A)|p_i} = \sum_{a \in A_i} a$ is regular. Moreover, let $p^F$ be the place generated by the folding construction. Then $M_{(F(N),A)}(p^F)$ defined by

$$M_{(F(N),A)}(p^F) = \sum_{(a_1,\ldots,a_n) \in A_1 \times \ldots \times A_n} (a_1, \ldots, a_n)$$

is the cartensian marking corresponding to $M_{(N,A)}(P^F)$. Moreover, $M_{(N,A)}(P^F)$ is the regular marking corresponding to $M_{(F(N),A)}(p^F)$.

**Proof:** Obviously the marking $M_{(F(N),A)}(p^F)$ is cartesian as defined in Def 5.2.9. $M_{(F(N),A)}(p^F) \in CP_{(F(N),A)}$ because $F(type)(p^F) = type(p_1) \star \ldots \star type(p_n)$ due to the definition of the folding construction and

$$A_1 \times \ldots \times A_n \subseteq A_{type(p_1)} \times \ldots \times A_{type(p_n)}$$

because $M_{(N,A)}(P^F)$ is regular. Vice versa let $M_{(F(N),A)}(p^F)$ be a cartesian marking, then we have $A_i \subseteq A_{type(p_i)}$ for $i \in \{1, \ldots, n\}$. Thus $M_{(N,A)}(P^F)$ is regular. □

In the following we show that the marking of a set of uniform places is not affected by the firing behavior because the pre- and post domain of these places are equal. Analogously, we show that the marking of a uniform place remains unchanged. We show this result especially for the uniform place generated by the folding construction. Here and in the following we use the indexes $(N, A)$ and $(F(N), A)$ to distinguish between the elements corresponding to these AHO-nets.

**Lemma 5.2.11 (Firing Steps wrt. Folding)**
Let $N = (\Sigma, P, T, pre, post, cond, type)$ be an AHO-net scheme, $P^F \subseteq P$ a set of uniform places with $P^F = \{p_1, \ldots, p_n\}$ and $F(N)$ the folding of $N$ wrt. product types and the set of uniform places $P^F$. Furthermore, let $(N, A)$ be an AHO-net and $M_{(N,A)}(P^F)$ be an arbitrary but fixed marking of the uniform places $P^F$ such that $M_{(N,A)|P^F}$ is regular and let $p^F$ is the place generated by the folding construction and $M_{(F(N),A)}(p^F)$ the cartesian marking corresponding to $M_{(N,A)}(P^F)$.

Given $M_{(N,A,P^F)} \in CP^{\oplus}_{(N,A,M(P^F))}$ and $(t, v) \in CT_{(N,A)}$ such that there is a firing step $M_{(N,A,P^F)}[(t, v)\rangle M'_{(N,A)}$, then we have

$$M'_{(N,A)} \in CP^{\oplus}_{(N,A,M(P^F))}.$$

Given $M_{(F(N),A,p^F)} \in CP^{\oplus}_{(F(N),A,M(p^F))}$ and $(t, v) \in CT_{(F(N),A)}$ such that there is a firing step $M_{(F(N),A,P^F)}[(t, v)\rangle M'_{(F(N),A)}$, then we have

$$M'_{(F(N),A)} \in CP^{\oplus}_{(F(N),A,M(P^F))}.$$

**Proof:** First we have show that $M'_{(N,A)|P^F} = M(P^F) = M_{(N,A,P^F)|P^F}$. Due to the definition of follower markings we have

$$M'_{(N,A)} = M_{(N,A,P^F)} \ominus \widehat{v}(pre(t)) \oplus \widehat{v}(post(t)).$$

We distinguish for $t \in T$ the following two cases:

**Case 1:** For $t \notin T^F$, i.e. the set of uniform places is not in the environment of $t$ we have

$$pre(t)_{|P^F} = post(t)_{|P^F} = \lambda.$$

Thus, $M_{(N,A,P^F)|P^F} = M'_{(N,A)|P^F}$.

**Case 2:** For $t \in T^F$, i.e. $t$ is in the pre- and post domain of $P^F$ we have

$$pre(t)_{|P^F} = post(t)_{|P^F}.$$

Thus, $M_{(N,A,P^F)|P^F} = M'_{(N,A)|P^F}$.

The proof of $M'_{(F(N),A)} \in CP^{\oplus}_{(F(N),A,M(P^F))}$ is more or less analog because we have for all $t \notin T^F$

$$F(pre)(t) = pre(t) \text{ and } F(post)(t) = post(t)$$

and for all $t \in T^F$

$$F(pre)(t)_{|p^F} = F(post)(t)_{|p^F}.$$

□

The second theorem in this section is divided into two main results. On the one hand the folding construction wrt. product types leads to equivalent AHO-nets concerning their firing behavior, and on the other hand the unfolding construction wrt. product types preserves the operational behavior. In detail, we show that for an arbitrary but fixed regular marking of the set of uniform places and for a corresponding cartesian marking of the uniform place the (un-)folded AHO-net has exactly the same sets of markings, consistent transitions valuations, enabled transitions, firing steps, and reachable markings as the AHO-net.

**Theorem 5.2.12 (Folding and Unfolding w.r.t. Product Types)**
Let $N$ be an AHO-net scheme with $N = (\Sigma, P, T, pre, post, cond, type)$ and $A$ a higher-order partial $\Sigma$-algebra.

1. Let $F(N)$ be the folding of $N$ wrt. product types and the set of uniform places $P^F \subseteq P$. Let $(N, A)$ be an AHO-net and $M_{(N,A)}(P^F)$ an arbitrary but fixed regular marking of the set of uniform places. Moreover, let $p^F$ be the place generated by the folding construction and $M_{(F(N),A)}(p^F)$ be the cartesian marking corresponding to $M_{(N,A)}(P^F)$. Then the AHO-net $(N, A)$ and the AHO-net $(F(N), A)$ are equivalent wrt. their firing behavior. More precisely,

   (a) the markings are in a bijective correspondence:
   $M_{(N,A,P^F)} \in CP^{\oplus}_{(N,A,M(P^F))} \iff M_{(F(N),A,p^F)} \in CP^{\oplus}_{(F(N),A,M(p^F))}$,

   (b) the sets of consistent transition valuations are in a bijective correspondence:
   $(t, v) \in CT_{(N,A)} \iff (t, v) \in CT_{(F(N),A)}$,

   (c) the set of enabled transition are equivalent:
   $M_{(N,A,P^F)}[(t, v)\rangle \iff M_{(F(N),A,p^F)}[(t, v)\rangle$,

   (d) the sets of firing steps are equivalent:
   $M_{(N,A,P^F)}[(t, v)\rangle M'_{(N,A)} \iff M_{(F(N),A,p^F)}[(t, v)\rangle M'_{(F(N),A)}$,

   (e) $CP^{\oplus}_{(N,A,M(P^F))}$ and $CP^{\oplus}_{(F(N),A,M(p^F))}$ are closed under reachability and the set of reachable markings are equivalent:
   $M'_{(N,A,P^F)} \in [M_{(N,A,P^F)}\rangle \iff M'_{(F(N),A,p^F)} \in [M_{(F(N),A,p^F)}\rangle$.

2. Let $U(N)$ be the unfolding of $N$ wrt. product types and the uniform place $p^U \in P$. Let $(N, A)$ be an AHO-net and let $M_{(N,A)}(p^U)$ an arbitrary but fixed cartesian marking of the uniform place. Let $P^U$ be the set of places generated by the unfolding construction and $M_{(U(N),A)}(P^U)$ be the regular marking corresponding to $M_{(U(N),A)}(p^U)$. Then the AHO-net $(N, A)$ and the AHO-net $(U(N), A)$ are equivalent wrt. their firing behavior. More precisely,

   (a) the markings are in a bijective correspondence:
   $M_{(N,A,p^U)} \in CP^{\oplus}_{(N,A,M(p^U))} \Longleftrightarrow M_{(U(N),A,P^U)} \in CP^{\oplus}_{(U(N),A,M(P^U))}$,

   (b) the sets of consistent transition valuations are in a bijective correspondence:
   $(t, v) \in CT_{(N,A)} \Longleftrightarrow (t, v) \in CT_{(U(N),A)}$,

   (c) the set of enabled transition are equivalent:
   $M_{(N,A,p^U)}[(t, v)\rangle \Longleftrightarrow M_{(U(N),A,P^U)}[(t, v)\rangle$,

   (d) the sets of firing steps are equivalent:
   $M_{(N,A,p^U)}[(t, v)\rangle M'_{(N,A)} \Longleftrightarrow M_{(U(N),A,P^U)}[(t, v)\rangle M'_{(U(N),A)}$,

   (e) $CP^{\oplus}_{(N,A,M(p^U))}$ and $CP^{\oplus}_{(U(N),A,M(P^U))}$ are closed under reachability and the set of reachable markings are equivalent:
   $M'_{(N,A,p^U)} \in [M_{(N,A,p^U)}\rangle \Longleftrightarrow M'_{(U(N),A,P^U)} \in [M_{(U(N),A,P^U)}\rangle$.

**Proof:** The proof is a consequence of the definition of the folding and unfolding constructions and the definition of the firing behavior of AHO-nets.

**1. (N,A) and (F(N),A) are equivalent wrt. their firing behavior.**
Let $P^F \subseteq P$ a set of uniform places with corresponding list

$$P^F = \langle p_1 : type_1, \ldots, p_n : type_n\rangle, n \geq 2$$

and pre- and post domain $T^F \subseteq T$. Let $p^F$ the place generate by the folding construction. Then we have due to the definition of the folding construction

$$(P \setminus P^F) = (F(P) \setminus \{p^F\}) \tag{5.5}$$

and for all $t \in T$:

$$F(cond)(t) = cond(t). \tag{5.6}$$

Moreover, we have for all $t \in T^F$:

$$pre_{|P^F}(t) = post_{|P^F}(t), \tag{5.7}$$

$$F(pre)(t)_{|(F(P)\setminus\{p^F\})} = pre(t)_{|(P\setminus P^F)}, \tag{5.8}$$

$$F(post)(t)_{|(F(P)\setminus\{p^F\})} = post(t)_{|(P\setminus P^F)}, \text{ and} \tag{5.9}$$

$$F(pre)(t)_{|p^F} = F(post)(t)_{|p^F} = \langle term_1, \ldots, term_n\rangle, \tag{5.10}$$

if $pre(t)_{|P^F} = \sum_{i=1}^{n}(term_i, p_i)$. For all $t \notin T^F$ we have

$$F(pre)(t) = pre(t), F(post)(t) = post(t) \text{ and} \tag{5.11}$$

$$pre(t)_{|P^F} = post(t)_{|P^F} = \lambda. \tag{5.12}$$

**(a)** First we define

$$M_{(N,A,P^F)|(P\setminus P^F)} = M_{(F(N),A,p^F)|(F(P)\setminus\{p^F\})}. \tag{5.13}$$

These markings are well-defined by (5.5). Hence

$$M_{(N,A,P^F)} \in CP^{\oplus}_{(N,A,M(P^F))} \iff M_{(F(N),A,p^F)} \in CP^{\oplus}_{(F(N),A,p^F)},$$

because by Lemma 5.2.10 $M_{(F(N),A)}(p^F)$ is the cartesian marking corresponding to $M_{(N,A)}(P^F)$ .

**(b)** It remains to show for all $t \in T$:

$$\begin{aligned}
& cond(t) \in dom(v^\sharp) \text{ and} \\
& \forall (term, p) \in pre(t) \oplus post(t) : term \in dom(v^\sharp) \\
\iff\ & F(cond)(t) \in dom(v^\sharp) \text{ and} \\
& \forall (term', p') \in F(pre)(t) \oplus F(post)(t) : term' \in dom(v^\sharp).
\end{aligned}$$

We have by (5.6) that

$$cond(t) \in dom(v^\sharp) \iff F(cond)(t) \in dom(v^\sharp).$$

Next we distinguish for $t \in T$ the following two cases:

**Case 1:** For $t \notin T^F$ it follows by (5.11) that

$$\begin{aligned}
& \forall (term, p) \in pre(t) \oplus post(t) : term \in dom(v^\sharp) \\
\iff\ & \forall (term', p') \in F(pre)(t) \oplus F(post)(t) : term' \in dom(v^\sharp).
\end{aligned}$$

**Case 2:** For $t \in T^F$ we have by (5.8) and (5.9)

$$\begin{aligned}
& \forall (term, p) \in pre(t)_{|(P\setminus P^F)} \oplus post(t)_{|(P\setminus P^F)} : term \in dom(v^\sharp) \\
\iff\ & \forall (term, p) \in F(pre(t))_{|(F(P)\setminus\{p^F\})} \oplus F(post(t))_{|(F(P)\setminus\{p^F\})} : \\
& \qquad\qquad term \in dom(v^\sharp)
\end{aligned}$$

and by (5.10)

$$\begin{aligned}
& \forall (term, p) \in pre(t)_{|P^F} \oplus post(t)_{|P^F} : term \in dom(v^\sharp) \\
\iff\ & \forall (term_1, p_1) \in pre(t)_{|p_1} \oplus pre(t)_{|p_1}, \ldots, \\
& \forall (term_n, p_n) \in pre(t)_{|p_n} \oplus pre(t)_{|p_n} : \\
& \qquad\qquad term_1 \in dom(v^\sharp), \ldots, term_n \in dom(v^\sharp) \\
\iff\ & \forall (term_1, p_1) \in pre(t)_{|p_1} \oplus pre(t)_{|p_1}, \ldots, \\
& \forall (term_n, p_n) \in pre(t)_{|p_n} \oplus pre(t)_{|p_n} : \\
& \qquad\qquad \langle term_1, \ldots, term_n \rangle \in dom(v^\sharp) \\
\iff\ & \forall (\langle term_1, \ldots, term_n \rangle, p^F) \in F(pre)(t)_{|p^F} \oplus F(post)(t)_{|p^F} : \\
& \qquad\qquad \langle term_1, \ldots, term_n \rangle \in dom(v^\sharp) \\
\iff\ & \forall (term', p^F) \in F(pre)(t)_{|p^F} \oplus F(post)(t)_{|p^F} : term' \in dom(v^\sharp).
\end{aligned}$$

Hence

$$\begin{aligned}
& \forall (term, p) \in pre(t) \oplus post(t) : term \in dom(v^\sharp) \\
\iff\ & \forall (term', p') \in F(pre)(t) \oplus F(post)(t) : term' \in dom(v^\sharp).
\end{aligned}$$

**(c)** It remains to show for all $t \in T$:

$$\widehat{v}(pre(t)) \leq M_{(N,A,P^F)} \iff \widehat{v}(F(pre)(t)) \leq M_{(F(N),A,p^F)}.$$

We distinguish for $t \in T$ the following two cases:

**Case 1:** For $t \notin T^F$ we have

$$\begin{aligned}
& \widehat{v}(pre(t)) \leq M_{(N,A,P^F)} \\
\iff\ & \widehat{v}(pre(t)) \leq M_{(N,A,P^F)|(P\setminus P^F)} \oplus M_{(N,A,P^F)|P^F} \\
\iff\ & \widehat{v}(pre(t)) \leq M_{(N,A,P^F)|(P\setminus P^F)} && \text{(by (5.12))} \\
\iff\ & \widehat{v}(F(pre)(t)) \leq M_{(F(N),A,p^F)|F(P)\setminus\{p^F\}} \\
& && \text{(by (5.11) and (5.13))} \\
\iff\ & \widehat{v}(F(pre)(t)) \leq M_{(F(N),A,p^F)|F(P)\setminus\{p^F\}} \oplus M_{(F(N),A,p^F)|p^F} \\
\iff\ & \widehat{v}(F(pre)(t)) \leq M_{(F(N),A,p^F)}.
\end{aligned}$$

**Case 2:** For $t \in T^F$ we have by (5.8) and (5.13)

$$\begin{aligned}
& \widehat{v}(pre(t)_{|(P\setminus P^F)}) \leq M_{(N,A,P^F)|(P\setminus P^F)} \\
\iff\ & \widehat{v}(F(pre)(t)_{|(F(P)\setminus\{p^F\})}) \leq M_{(F(N),A,p^F)|(F(P)\setminus\{p^F\})}.
\end{aligned}$$

By Lemma 5.2.10 $M_{(F(N),A}(p^F)$ is the cartesian marking corresponding to $M_{(N,A}(P^F)$ and by (5.10) it follows that

$$\begin{aligned}
& \widehat{v}(pre(t)_{|P^F}) \leq M_{(N,A,P^F)|P^F} \\
\iff\ & \widehat{v}(F(pre)(t)_{|p^F}) \leq M_{(F(N),A,p^F)|p^F}.
\end{aligned}$$

Therefore

$$\widehat{v}(pre(t)) \leq M_{(N,A,P^F)} \iff \widehat{v}(F(pre)(t)) \leq M_{(F(N),A,p^F)}.$$

**(d)** Given $M_{(N,A,P^F)}[(t,v)\rangle$ and $M_{(F(N),A,p^F)}[(t,v)\rangle$, such that

$$M_{(N,A,P^F)}[(t,v)\rangle \iff M_{(F(N),A,p^F)}[(t,v)\rangle,$$

due to the definition of the follower markings $M'_{(N,A)}$ and $M'_{(F(N),A)}$, we have to show that

$$\begin{aligned}
& M'_{(N,A)} = M_{(N,A,P^F)} \ominus \widehat{v}(pre(t)) \oplus \widehat{v}(post(t)) \\
\iff\ & M'_{(F(N),A)} = M_{(F(N),A,p^F)} \ominus \widehat{v}(F(pre)(t)) \oplus \widehat{v}(F(post)(t)).
\end{aligned}$$

By Lemma 5.2.11 we have

$$M'_{(N,A)} \in CP^{\oplus}_{(N,A,M(P^F))} \text{ and } M'_{(F(N),A)} \in CP^{\oplus}_{(F(N),A,M(P^F))}.$$

We distinguish for $t \in T$ the following two cases:

**Case 1:** For $t \notin T^F$ the equivalence follows immediately by (5.11), (5.12) and Proof of **1. (a)**.

**Case 2:** For $t \in T^F$ we have by (5.8), (5.9), and (5.13)

$$\begin{aligned}
& M'_{(N,A)|(P\setminus P^F)} = M_{(N,A,P^F)|(P\setminus P^F)} \\
& \qquad \ominus \widehat{v}(pre(t)_{|(P\setminus P^F)}) \oplus \widehat{v}(post(t)_{|(P\setminus P^F)}) \\
\iff\ & M'_{(F(N),A)|(F(P)\setminus\{p^F\})} = M_{(F(N),A,p^F)|(F(P)\setminus\{p^F\})} \\
& \quad \ominus \widehat{v}(F(pre)(t)_{|(F(P)\setminus\{p^F\})}) \oplus \widehat{v}(F(post)(t)_{|(F(P)\setminus\{p^F\})}).
\end{aligned}$$

$M_{(F(N),A)}(p^F)$ is the cartesian marking corresponding to $M_{(N,A)}(P^F)$ and by (5.10) we have

$$\begin{aligned} & M'_{(N,A)|P^F} = M_{(N,A,P^F)|P^F} \\ & \qquad \ominus\widehat{v}(pre(t)_{|P^F}) \oplus \widehat{v}(post(t)_{|P^F}) \\ \iff \quad & M'_{(F(N),A)|p^F} = M_{(F(N),A,p^F)|p^F} \\ & \qquad \ominus\widehat{v}(F(pre)(t)_{|p^F}) \oplus \widehat{v}(F(post)(t)_{|p^F}). \end{aligned}$$

Therefore

$$\begin{aligned} & M'_{(N,A)} = M_{(N,A(N,A,P^F))} \ominus \widehat{v}(pre(t)) \oplus \widehat{v}(post(t)) \\ \iff \quad & M'_{(F(N),A)} = M_{(F(N),A,p^F)} \ominus \widehat{v}(F(pre)(t)) \oplus \widehat{v}(F(post)(t)). \end{aligned}$$

**(e)** By induction over the length of occurrence sequences that $CP^{\oplus}_{(N,A,M(P^F))}$ and $CP^{\oplus}_{(F(N),A,M(p^F))}$ are closed under reachability and the set of reachable markings are equivalent. For $n = 0$ it follows immediately due to the Proof of **(a)** that

$$M_{(N,A,P^F)} \in [M_{(N,A,P^F)}\rangle \iff M_{(F(N),A,p^F)} \in [M_{(F(N),A,p^F)}\rangle.$$

For $n = 1$ there are firing steps

$$M_{(N,A,P^F)}[(t,v_1)\rangle M'_{(N,A)} \text{ and } M_{(F(N),A,p^F)}[(t,v_2)\rangle M'_{(F(N),A)}$$

due to Proof of **(d)**. By Lemma 5.2.11 we have

$$M'_{(N,A)} \in CP^{\oplus}_{(N,A,M(P^F))} \text{ and } M'_{(F(N),A)} \in CP^{\oplus}_{(F(N),A,M(P^F))}.$$

Hence,

$$M'_{(N,A)} \in [M_{(N,A,P^F)}\rangle \iff M'_{(F(N),A)} \in [M_{(F(N),A,p^F)}\rangle.$$

For occurrence sequences of the length $n+1$ it follows due to the induction base and the induction hypothesis that for the end markings we have:

$$M'_{(N,A),n+2} \in CP^{\oplus}_{(N,A,M(P^F))} \text{ and } M'_{(F(N),A),n+2} \in CP^{\oplus}_{(F(N),A,M(P^F))}$$

and

$$M_{(N,A),n+2} \in [M_{(N,A,P^F)}\rangle \iff M_{(F(N),A),n+2} \in [M_{(F(N),A,p^F)}\rangle.$$

**2. (N,A) and (U(N),A) are equivalent wrt. their firing behavior.**
Let $N = (\Sigma, P, T, pre, post, cond, type)$ be an AHO-net scheme and $U(N)$ be the unfolding wrt. product types of $N$. Moreover, let $F$ be a folding construction such that $F(U(N)) \cong N$ (see Thm. 5.2.7). We define $U(N) = N'$. Then we have due to Proof of **1.(a)**

$$\begin{aligned} & (M_{(N',A,P^F)} \in CP^{\oplus}_{(N',A,M(P^F))} \iff M_{(F(N'),A,p^F)} \in CP^{\oplus}_{(F(N'),A,M(p^F))}) \\ \implies \quad & (M_{(U(N),A,P^F)} \in CP^{\oplus}_{(U(N),A,M(P^F))} \\ & \qquad \iff M_{(F(U(N)),A,p^F)} \in CP^{\oplus}_{(F(U(N)),A,M(p^F))}) \\ \implies \quad & (M_{(U(N),A,P^U)} \in CP^{\oplus}_{(U(N),A,M(P^U))} \iff M_{(N,A,p^U)} \in CP^{\oplus}_{(N,A,M(p^U))}) \end{aligned}$$

because $P^F = P^U$ and $p^F = p^U$. Analogously for **2.(b) - (e)**. □

# Chapter 6

# AHO-Nets for Specific Applications Domains

In this chapter we motivate the notions and results of this thesis from a practical point of view. The general idea is to introduce Petri nets and rules as tokens leading to three particular algebraic higher-order net classes, each of them for specific application domains. While in the previous chapters algebraic higher-order nets are considered with an arbitrary but fixed higher-order signature and corresponding higher-order algebra, in this chapter we provide a specific data type part for each particular net class. The goal of this chapter is manyfold. By means of several large examples we improve not only the comprehensibility of our new concept, but show also exemplarily some relations to existing net classes, particularly to elementary object nets [Val98, Val01] and algebraic high-level nets [PER95, Pad96, EHP+02]. Furthermore, we provide several basis formalisms like reconfigurable nets for further research and deeper analysis of important aspects.

In Section 6.1 we investigate algebraic higher-order net and interaction relation systems, which are motivated by the concepts of elementary object nets [Val98, Val01]. Afterwards, in Section 6.2, algebraic higher-order net and rule systems are presented in the spirit of the example of algebraic high-level nets from Section 3.3. Finally, in Section 6.3 we introduce the approach of algebraic higher-order rule systems, especially designed for application domains of mobile policies.

## 6.1 AHO-Net and Interaction Relation Systems

Here we review the concept of elementary object systems. Then we capture this concept in our approach and define a specific class of AHO-nets, called algebraic higher-order net and interaction systems. Moreover, we show exemplarily, that elementary object systems can be translated into semantically equivalent algebraic higher-order net and interaction systems.

High-level net models following the paradigm "nets as tokens" have been already a subject in the literature with several interesting applications. The paradigm "nets as tokens" has been introduced by Valk to allow nets as tokens, called object nets, within a net, called system net (see [Val98, Val01]). This paradigm has been very useful to model applications in the area of workflow, agent-oriented approaches, or flexible manufacturing systems. In elementary object systems, object nets can move through a system net and interact with both the system net and with other object nets. This may change the marking not only of the system net but also of object nets. The approach of elementary object systems has its origins in [JV87, Val87, Val91], where especially for the area of workflows so-called task systems are

introduced. Task systems are more complex than ordinary workflows because the system net reflects the organizational structure of the system and there may be different workflows (object nets) for the same system.

Elementary object systems allow a simple notion of nets as tokens. Most principles of elementary net theory are respected and extended. Elementary object systems are composed of a system net and one or more object nets, where both the system net and the object nets are considered as elementary net systems. Moreover, there is a special component of indistinguishable tokens. Usually such tokens are used for synchronization and modeling of resources. Thus places in elementary object systems can be marked either by a set of object systems or by a number of tokens.

In elementary object systems there is not only communication between the system net and object nets, but also communication between different object nets. This is formalized by a so-called interaction relation, which consists of a system-object interaction relation and an object-object interaction relation. In detail, the system-object interaction relation is defined by a relation of system net transitions and object net transitions which have to be fired in parallel, while the object-object interaction relation is a relation of object net transitions and guards the parallel firing of transitions in different object nets.

Elementary object systems do not reflect fork/join-control structures correctly (see [Val98] for a simple example). A fork-control structure can be understood as a creation or duplication of object nets, while a join-control structure reflects some kind of merging or destroying of object nets. But this would lead to inconsistencies in the definition of the dynamical behavior of elementary object systems. Thus elementary object systems have to be structural state machines, i.e. each transition of the system net has exactly one input place and exactly one output place and each object net occurs exactly once in an elementary object system. In this way the creation and destroying of object nets is forbidden in elementary object systems.

For a detailed definition of elementary object systems we refer to Appendix C, while in the following we focus on the firing behavior of elementary object systems to improve the comprehensibility of elementary object systems.

There are mainly four different kinds of occurrence rules, which are exemplarily depicted in Fig. 6.1 -Fig. 6.4. For the special component of indistinguishable tokens elementary object systems behave like ordinary P/T-systems. The object net $(ON_i, m_i)$ of Fig. 6.1 should be seen as tokens in place $p$ of the system net $SN$. Here and in the following we use zoom lines to illustrate the general distinction between the system net and the object nets. The zoom lines enlarge the object net $(ON_i, m_i)$ which is represented by a token in the place $p$. The term $(i)$ is an element of the arc inscription, i.e. the object $(ON_i, m_i)$ can be moved along the arc. Arc inscriptions can contain more values than $(i)$ as denoted by $\ldots + (i) + \ldots$. But be aware that thess arc inscriptions are interpreted in a such way that the object nets, which correspond to the values, have the possibility to move but not all of them have to move simultaneously. Since the transition $t$ of the system net $SN$ has no label, the object net $(ON_i, m_i)$ is moved to $p'$ by the occurrence of transition $t$. Because it does not change the marking of the object net, such an occurrence is called system-autonomous.

In a dual sense transition $e_1$ of $(ON_i, m_i)$ can also occur without interacting with the system or other object nets (see Fig. 6.2). Therefore such an occurrence is called object-autonomous.

An example of a system-object interaction occurrence rule is depicted in Fig. 6.3. Both nets $(ON_i, m_i)$ and $SN$ have reached a marking where $e_1$ and $t$ are activated. Since they have the same label (<i1> in this case) they must occur simultaneously, i.e. the pair $(t, e_1)$ is in the corresponding system-object interaction relation. Thus, the object net $(ON_i, m_i)$ is removed from place $p$ and the object net $(ON_i, m_i')$ is

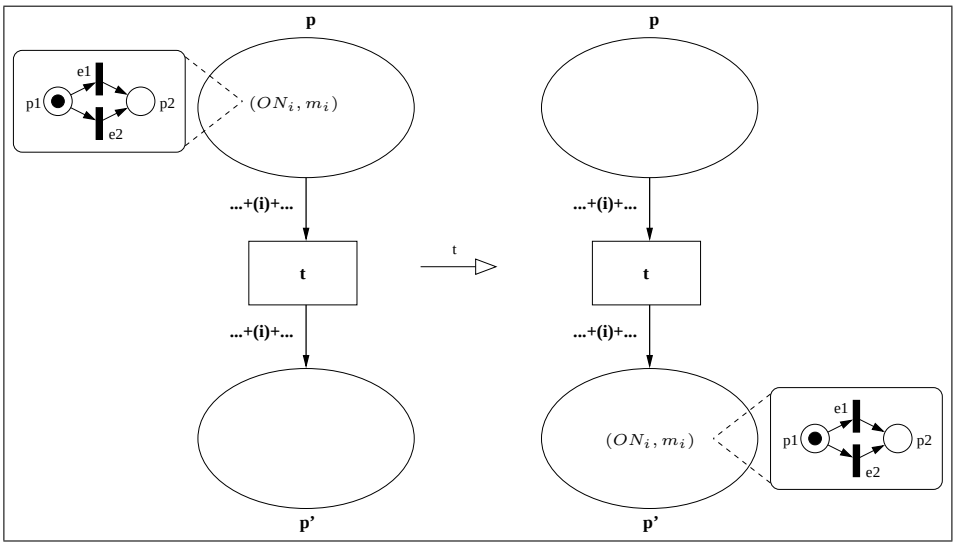



Figure 6.1: System-autonomous occurrence rule

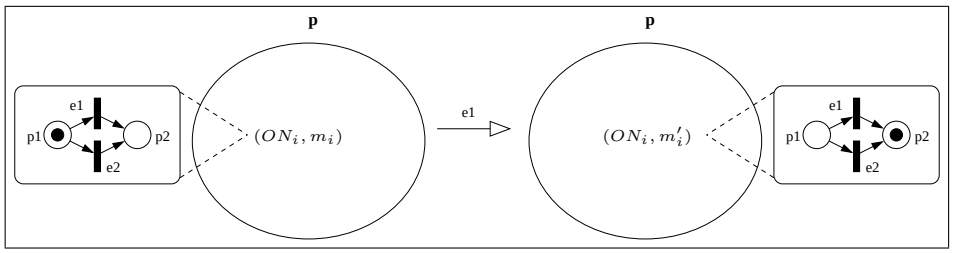



Figure 6.2: Object-autonomous occurrence rule

added to the place $p'$, where $m_i'$ is the follower marking of the object net $(ON_i, m_i)$.

Finally, Fig. 6.4 shows an object-object interaction occurrence rule. Here object nets $(ON_i, m_i)$ and $(ON_j, m_j)$ have both reached a marking where $ei1$ and $ej1$ are activated. As indicated by the same label ([r] in this case), the pair $(ei1, ej1)$ is in the corresponding object-object interaction relation. Thus, these transitions must occur simultaneously, i.e. the follower markings $m_i'$ and $m_j'$ are computed. But in contrast to the system-object interaction occurrence rule the object nets still remain on the place $p$.

**Example 6.1.1 (Hurried Philosophers as Elementary Object System)** Especially the concept of elementary object systems [Val01] has been used to model the case study of the hurried philosophers proposed in [SB01].

Consider the system net $SN$ in Fig. 6.5, which consists of two places for the dining room and the library and two transitions to leave and enter the dining room. Initially there are five philosophers $phi_1, \ldots, phi_5$ in the library. The object nets of the philosophers are shown in Fig. 6.6, in full detail for the token net of philosopher $phi_1$ and in part for his/her right neighbor $phi_2$. The fork exchange is modeled by the transitions *request left neighbor*, *request right neighbor*, *grant left*, and *grant right*. The partners of the fork exchange are fixed, especially we have $phi_1$ as the right neighbour of $phi_5$.

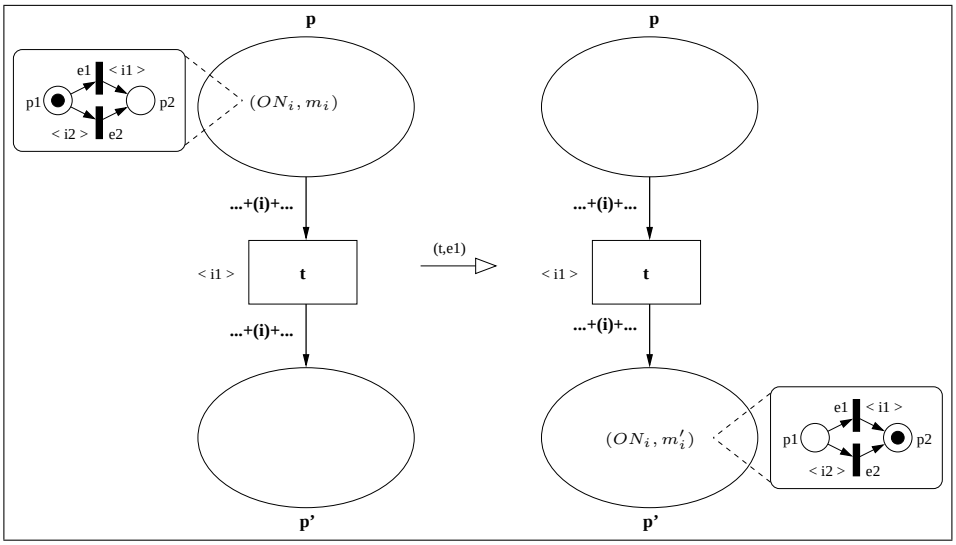



Figure 6.3: System-object interaction occurrence rule

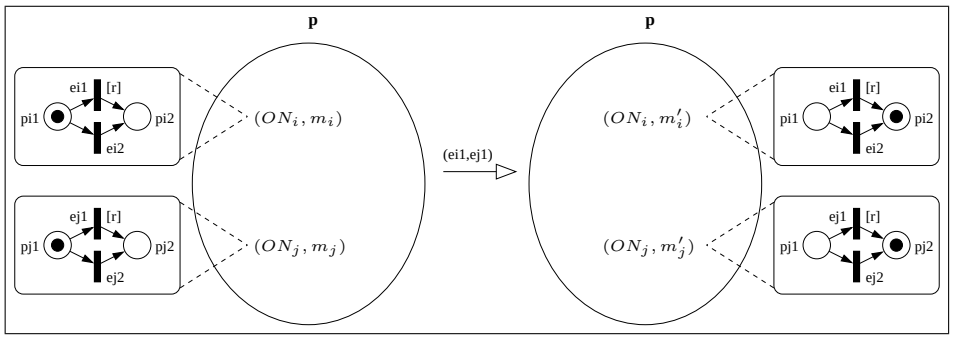



Figure 6.4: Object-object interaction occurrence rule

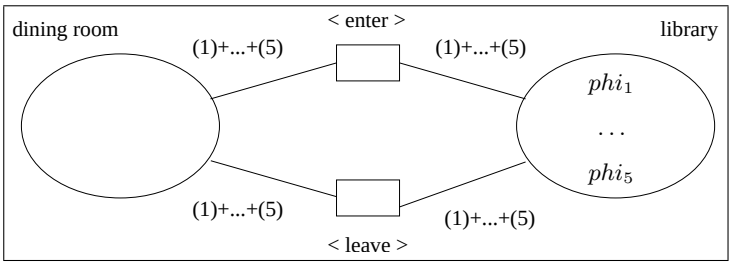



Figure 6.5: "Hurried philosophers" as elementary object system

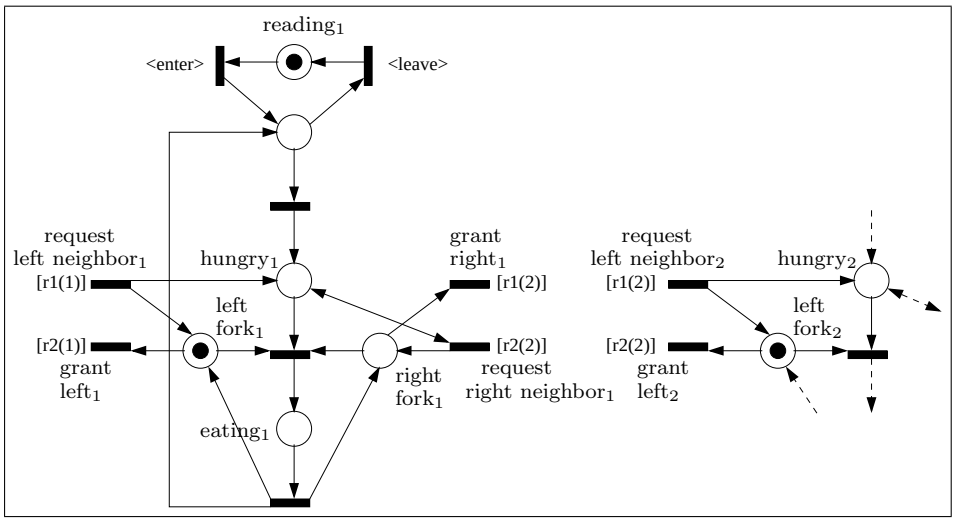



Figure 6.6: Object nets of philosopher $phi_1$ and his/her neighbor

Philosophers can move into the dining room due to the firing of the transitions $<enter>$. Assume philosopher $phi_1$ wants to enter the dining room. The label indicates that there is a communication between the system net $SN$ and the object net $phi_1$, i.e. the corresponding transition fires simultaneously and philosopher $phi_1$ arrives at the dining room with his/her left fork in the hand.

To start the fork exchange philosopher $phi_1$, being hungry, can borrow the missing right fork from his neighbor $phi_2$ (if he is also in the dining room) by interaction of the transitions labeled with [r2(2)], i.e. there is an object-object communication between $phi_1$ and $phi_2$ to give up his/her left fork. All neighboring philosophers can exchange the shared fork in the same way provided that they are in the dining room.

Many different settings of the distributed philosophers can be realized, e.g. the philosophers can only communicate by sending messages or a fork shuttle could move around and distribute forks to arbitrary participants. But the expressive power of elementary object systems is restricted in the sense, that the updating of the seating arrangement during run time, i.e. the seating arrangement might be permanently modified, can not be modeled because the partners of the fork exchange are fixed once and for all in the corresponding interaction relation.

By contrast to elementary object systems, in our approach of AHO-nets we use different formal frameworks for the system level and the object level. In detail the object level is specified in the data type part, while the system level is modeled by AHO-nets. Thus in order to capture the concept of elementary object systems, we first define a suitable signature NI-SIG and a corresponding NI-SIG-algebra in our approach.

**Definition 6.1.2 (NI-SIG Signature)**
The signature NI-SIG for elementary net systems with interaction relations as tokens is given by

NR-SIG =

sorts: $Token, Transitions, Places, ENSystem, IntRel$

opns: $tt$, $ff$: $Pred(unit)$

$fire : ((ENSystem \star IntRel \star Transitions) \rightarrow (ENSystem \star IntRel))$

$isInSysObj : Pred(ENSystem \star IntRel \star Transitions)$

$isInObj : Pred(ENSystem \star IntRel \star Transitions)$

$isInObjObj : Pred(ENSystem \star IntRel \star ENSystem \star IntRel \star Transitions \star Transitions)$

and for each function type $(type_1 \rightarrow type_2) \in S^{\rightarrow}$ we have an application symbol $apply^{type_1,type_2} : (type_1 \rightarrow type_2) \star type_1 \rightharpoonup type_2$.

For a detailed notion of elementary net (EN) systems we refer to Appendix C.1.

**Definition 6.1.3 (NI-SIG Algebra)**
Given vocabularies $E_0$ and $B_0$, the carrier of the NI-SIG-algebra $A$ for EN-systems with interaction relation as tokens is given by

- $A_{Token} = \{\bullet\}$,
- $A_{Transitions} = E_0$, $A_{Places} = B_0$,
- $A_{Pred(unit)} = \{true, false\}$,
- $A_{ENSystem}$ the set of all EN-systems over $E_0$ and $B_0$, i.e.
  $A_{ENSystem} = \{EN | EN = (B, E, pre, post, m)$ EN-system, $B \subseteq B_0, E \subseteq E_0\}$,
- $A_{IntRel} = \{(E^{SO}, E^O, E^{OO}) | (E^{SO} \cup E^O) \subseteq E_0, E^{SO} \cap E^O = \emptyset,$
  $E^{OO} \subseteq (E_0 \times E_0) \setminus id_{E_0}, E^{OO}$ symmetric,
  $(e, e') \in E^{OO} \Rightarrow e, e' \notin (E^{SO} \cup E^O)\}$,
- $A_{(ENS \star IR \star TR) \rightarrow (ENS \star IR)} = \{fire\}$,
- $A_{Pred(ENS \star IR \star TR)} = \{isInSysObj, isInObj\}$, and
- $A_{Pred(ENS \star IR \star ENS \star IR \star TR \star TR)} = \{isInObjObj\}$

where here and in the following $TR$ is an abbreviation for $Transitions$, $ENS$ for $ENSystems$, and $IR$ for $IntRel$.
For all $type \in BFS^{\rightarrow}$ different from the types above we have

$$A_{type} = \emptyset.$$

The carrier is extended to the set of higher-order types $S^{\rightarrow}$ by

$$A_{unit} := \{()\} \text{ and}$$
$$A_{type_1 \star \ldots \star type_n} := A_{type_1} \times \ldots \times A_{type_n}.$$

The partial operations of the NI-SIG-algebra $A$ are defined by

- $tt_A = true$, $ff_A = false$, and
  $apply_A^{unit,unit} : \{true, false\} \times A_{unit} \multimap A_{unit}$
  with
  $$(x.())^A = \begin{cases} () & \text{if } x = true \\ undef & \text{if } x = false \end{cases}$$

- $fire_A = fire$ and
  $apply_A^{(ENS \star IR \star TR),(ENS \star IR)} : \{fire\} \times (A_{ENS} \times A_{IR} \times E_0) \multimap (A_{ENS} \times A_{IR})$
  for $EN = (B, E, pre, post, m) \in A_{ENS}$, $Rel \in A_{IR}$, and $e \in E_0$ with

$$(fire.(EN, Rel, e))^A = \begin{cases} ((B, E, pre, post, (m \setminus pre(e)) \cup post(e)), Rel) \\ \qquad \text{if } e \in E, pre(e) \subseteq m \text{ and } post(e) \subseteq (B \setminus m) \\ undef \quad \text{else} \end{cases}$$

- $isInSysObj_A = isInSysObj$, $isInObj_A = isInObj$, and
  $apply_A^{(ENS \star IR \star TR),unit}$ :
  $\{isInSysObj, isInObj\} \times (A_{ENS} \times A_{IR} \times E_0) \multimap A_{unit}$
  for $EN = (B, E, pre, post, m) \in A_{ESystem}$, $Rel = (E^{SO}, E^O, E^{OO}) \in A_{IntRel}$, and $e \in E_0$ with

$$(isInSysObj.(EN, Rel, e))^A = \begin{cases} () & \text{if } e \in E, e \in E^{SO} \\ undef & \text{else} \end{cases}$$

  and

$$(isInObj.(EN, Rel, e))^A = \begin{cases} () & \text{if } e \in E, e \in E^{O} \\ undef & \text{else} \end{cases}$$

- $isInObjObj_A = isInObjObj$ and
  $apply_A^{(ENS \star IR \star ENS \star IR \star TR \star TR),unit}$ :
  $\{isInObjObj\} \times (A_{ENS} \times A_{IR} \times A_{ENS} \times A_{IR} \times E_0 \times E_0) \multimap A_{unit}$
  for $EN_i = (B_i, E_i, pre_i, post_i, m_i) \in A_{ENS}$, $Rel_i = (E_i^{SO}, E_i^O, E_i^{OO}) \in A_{IR}$, $e_i \in E_0$ for $i \in \{1, 2\}$ with

$$(isInOBjObj.(EN_1, Rel_1, EN_2, Rel_2, e_1, e_2))^A$$

$$= \begin{cases} () & \text{if } e_1 \in E_1, e_2 \in E_2, (e_1, e_2) \in E_1^{OO}, (e_1, e_2) \in E_2^{OO} \\ undef & \text{else} \end{cases}$$

Because there is no equivalent notion in the operational behavior of AHO-nets for the object-autonomous behavior and object-object communication as given in elementary object nets, in the following definition of algebraic higher-order net and interaction systems we have to introduce for each place one transition to simulate the object-autonomous behavior and one transition to simulate the object-object communication. Moreover, the object nets are equipped by the same interaction relation.

**Definition 6.1.4 (AHO-Net and Interaction Systems)**
Given the signature NI-SIG and the NI-SIG-algebra $A$ as above, an algebraic higher-order net and interaction system $AHONIS = ((N, A), INIT)$ consists of an AHO-net $(N, A)$ (see Def. 3.2.7) with $\Sigma = (\text{NI-SIG}, X)$ where $X$ are variables over NI-SIG and an initial marking $INIT$ so that

- for $X$ we have
  - $X_{Token} = \{x\}$,
  - $X_{Transitions} = \{e, e_1, e_2\}$,

  - $X_{ENSystem} = \{sys, sys_1, sys_2\}$, and
  - $X_{IntRel} = \{rel, rel_1, rel_2\}$,

- all places $p \in P$ are either
  - EN-system and interaction relation places $p \in P_{ENSys} = \{p \in P | type(p) = (ENSystem \star IntRel)\}$ or
  - token places $p \in P_{Tok} = \{p \in P | type(p) = Token\}$,

- all transitions $t \in T$ are either
  - system-autonomous transitions $t \in T^S$ with
    $pre(t) = ((sys, rel), p_1) \oplus ((n \cdot x), p'_1)$,
    $post(t) = ((sys, rel), p_2) \oplus ((m \cdot x), p'_2)$, and
    $cond(t) = tt$
    where $p_1, p_2 \in P_{ENSys}$, $p'_1, p'_2 \in P_{Tok}$ and $m, n \in \mathbb{N}$,
  - object-autonomous transitions $t \in T^O$ with
    $pre(t) = ((sys, rel), p)$,
    $pre(t) = ((fire.(sys, rel, e)), p)$, and
    $cond(t) = (isInObj.(sys, rel, e))$
    where $p \in P_{ENSys}$,
  - system-object interaction transitions $t \in T^{SO}$ with
    $pre(t) = ((sys, rel), p_1) \oplus ((n \cdot x), p'_1)$,
    $post(t) = ((fire.(sys, rel, e)), p_2) \oplus ((m \cdot x), p'_2)$, and
    $cond(t) = (isInSysObj.(sys, rel, e))$
    where $p_1, p_2 \in P_{ENSys}$, $p'_1, p'_2 \in P_{Tok}$, and $m, n \in \mathbb{N}$, or
  - object-object interaction transitions $t \in T^{OO}$ with
    $pre(t) = ((sys_1, rel_1), p) \oplus ((sys_2, rel_2), p)$,
    $post(t) = ((fire.(sys_1, rel_1, e_1)), p) \oplus ((fire.(sys_2, rel_2, e_2)), p)$, and
    $cond(t) = (isInObjObj.(sys_1, rel_1, sys_2, rel_2, e_1, e_2))$
    where $p \in P_{ENSys}$,

  and there is a bijective correspondence

$$P_{ENSys} \cong T^O \cong T^{OO},$$

- and $INIT \in (A \otimes P)^{\oplus}$ with $INIT_{|P_{ENSys}} = \sum_{i=1}^{n}((ON_i, Rel), p_i)$ in normal form such that for $ON_i = (B_i, E_i, pre_i, post_i, m_i)$ for $i \in \{1, \ldots, n\}$ and $Rel = (E^{SO}, E^O, E^{OO})$ we have

$$E^{SO} \subseteq \mathbf{E}, E^O \subseteq \mathbf{E}, \text{ and } E^{OO} \subseteq \mathbf{E} \times \mathbf{E} \setminus id_{\mathbf{E}}.$$

  where $\mathbf{E} := \bigcup_{i \in \{1 \ldots, n\}} E_i$.

In the following we give for each occurrence rule of elementary object nets depicted in Fig. 6.1 - Fig. 6.4 an equivalent occurrence rule of AHONI-systems in Fig. 6.7 - Fig. 6.10. For the special component of indistinguishable tokens AHONI-systems behave like ordinary P/T-systems.

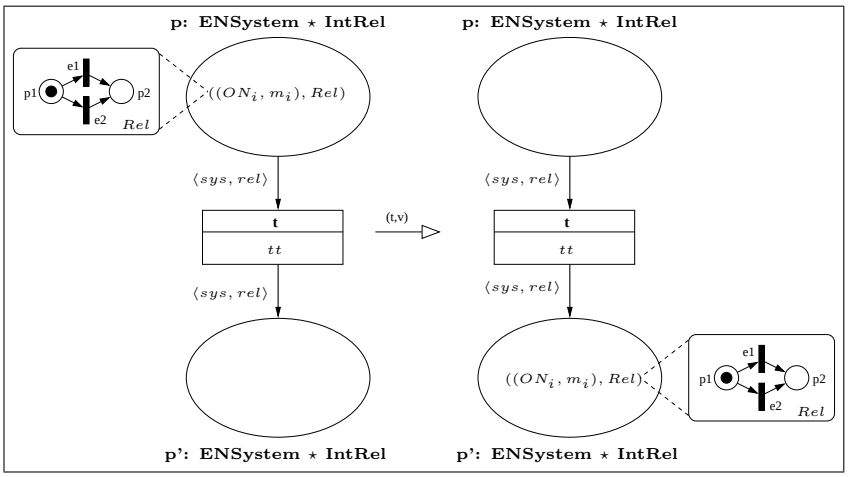



Figure 6.7: Occurrence rule for system-autonomous transitions

To fire the system-autonomous transition $t$ in Fig. 6.7, we have to give a variable valuation $v : Var(t) \longrightarrow A$, i.e. $v(sys, rel) = ((ON_i, m_i), Rel)$. Then the object net $((ON_i, m_i), Rel)$ is moved to $p'$ by the occurrence of transition $t$.

The object-autonomous transition (see Fig. 6.8) and the system-object interaction transition (see Fig. 6.9) have the same net inscriptions in their pre and post domain but they differ on the one hand in their firing conditions and on the other hand in the places in their pre and post domain. While the object-autonomous transition has to have the same place in its pre and post domain, the places of the system-object interaction transition may be different. However, to fire these transitions we have to give a variable valuation as above, but we have to give also a valuation for the variable $e$ so that $v(e)$ is a transition of the object net $((ON_i, m_i), Rel)$, which is enabled in the marking $m_i$, for instance $v(e) = e1$. Otherwise the evaluation of the term $(fire.\langle sys, rel, e\rangle)$ is undefined and the AHONI-systems is not enabled. Finally, we have to check the definedness of firing conditions. Let $Rel = (E^{SO}, E^{O}, E^{OO})$. In case of the object-autonomous transition $v(e)$ has to be an element of the relation $E^{O}$. In case of the system-object interaction transition $v(e)$ has to be an element of the relation $E^{SO}$. Then the follower marking of the object net $(ON_i, m_i)$ is computed due to the evaluation of the net inscription $\hat{v}((fire.\langle sys, rel, e\rangle)) = (fire.((ON_i, m_i), Rel, e1))^A = ((ON_i, m_i'), Rel)$.

Finally, Fig. 6.10 shows an occurrence rule for an object-object interaction transition where both $(ON_i, m_i)$ and $(ON_j, m_j)$ have reached a marking where $ei1$ and $ej1$ are activated. Thus we give a variable valuation $v(sys_1, rel) = ((ON_i, m_i), Rel)$, $v(sys_2, rel) = ((ON_j, m_j), Rel)$, $v(e_1) = ei1$, and $v(e_2) = ej1$. Moreover, for $Rel = (E^{SO}, E^{O}, E^{OO})$ $(e1i, ej1)$ has to be an element of the corresponding object-object interaction relation $E^{OO}$ as indicated by the firing condition. Thus the transitions $ei1$ and $ej1$ of the object nets and the object-object interaction transition $t$ occur simultaneously, i.e. the follower markings $m_i'$ and $m_j'$ are computed and the object nets still remain on the place $p$.

**Example 6.1.5 ("Hurried Philosophers" as AHONI-systems)**
In the following we use AHONI-systems to model the case study of the "Hurried Philosophers" [SB01]. The system level depicted in Fig. 6.11 consists of two places, *dining room* and *library*, but in contrast to the elementary object net in Fig. 6.5 there are in addition transitions to model on the one hand the object-object commu-

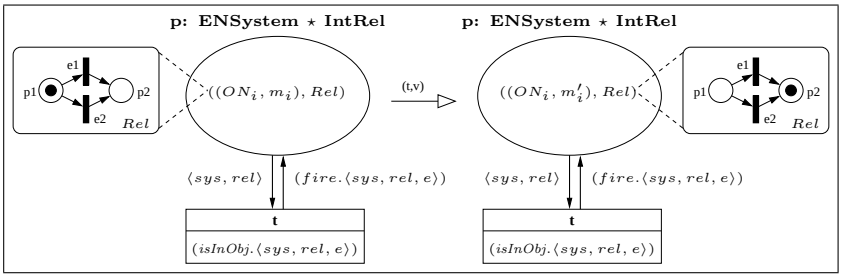



Figure 6.8: Occurrence rule for object-autonomous transitions

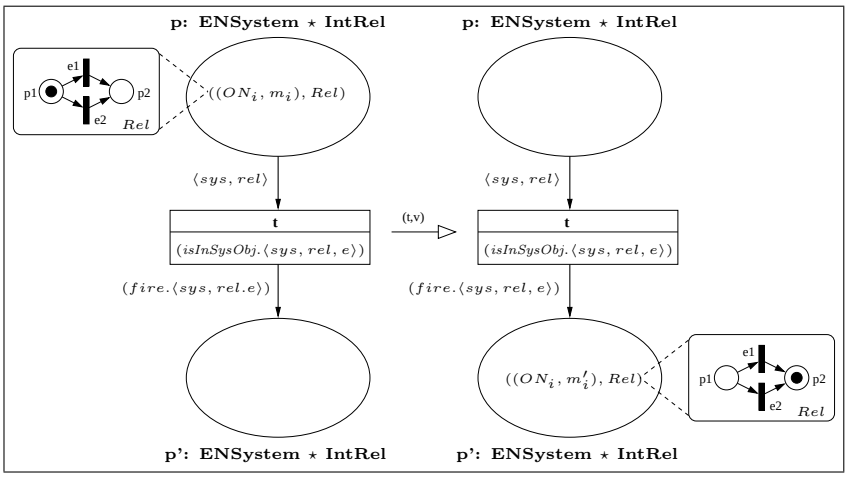



Figure 6.9: Occurrence rule for system-object interaction transitions

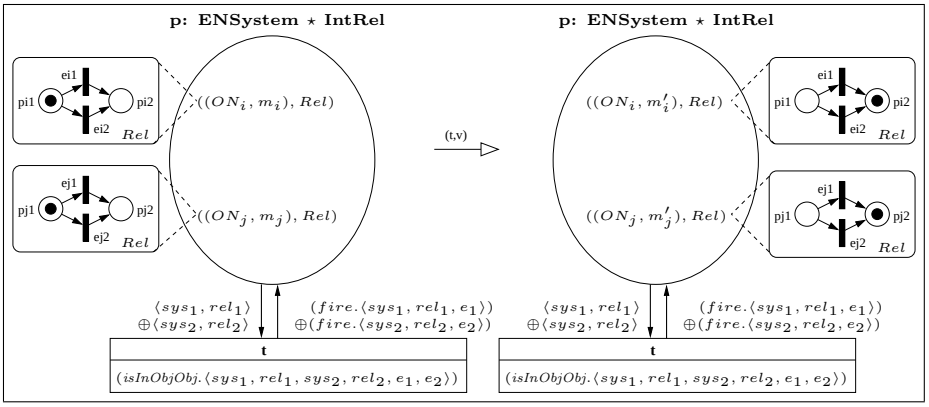



Figure 6.10: Occurrence rule for object-object interaction transitions

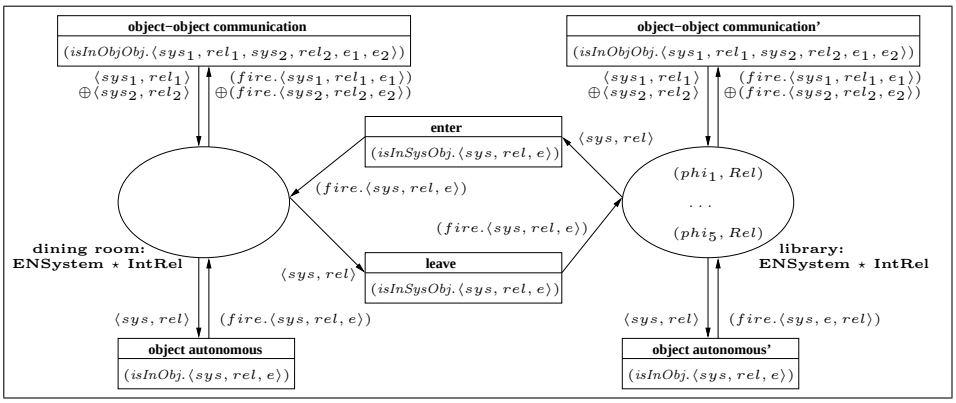



Figure 6.11: "Hurried Philosophers" as AHONI-system

nication and on the other hand the object autonomous behavior. Initially there are five philosophers $(phi_1, Rel), \ldots, (phi_5, Rel)$ in the library, where each philosopher is equipped with the same interaction relation $Rel$. The object net of the philosopher $phi_1$ is shown in Fig. 6.12. The interaction relation $Rel = (E^{SO}, E^O, E^{OO})$ is defined as follows.

$$
\begin{aligned}
E^{SO} &= \{enter_i, leave_i | i \in \{1, \ldots, 5\}\},\\
E^O &= \{t_{i1}, t_{i2}, t_{i3} | i \in \{1, \ldots, 5\}\}, \text{ and}\\
E^{OO} &= \{(\textit{request right neighbour}_k, \textit{grant left}_{k+1}),\\
&\qquad (\textit{grant left}_{k+1}, \textit{request right neighbour}_k),\\
&\qquad (\textit{request left neighbour}_{k+1}, \textit{grant right}_k),\\
&\qquad (\textit{grant right}_k, \textit{request left neighbour}_{k+1}, )\\
&\qquad | i \in \{1, \ldots, 5\}, k = (i+1) \ mod \ 5\}
\end{aligned}
$$

Philosophers move into the dining room by the firing of the transition *enter*. Assuming philosopher $phi_1$ wants to enter the dining room, we define a variable valuation $v : Var(enter) \longrightarrow A$ with $v(sys, rel) = (phi_1, Rel)$ and $v(e) = enter_1$. Because the transition $enter_1$ is activated in the object net of philosopher 1 and $enter_1$ is an element of the system-object interaction relation $E^{SO}$, the transition *enter* of the system net and the transition $enter_1$ of the object net are firing simultaneously and philosopher $phi_1$ arrives at the dining room with his/her left fork in the hand.

To start the fork exchange philosopher $phi_1$ can borrow the missing right fork from his neighbor $phi_2$, being in the dining room, by interaction of the transitions *request right neighbour*$_1$ in the object net $phi_1$ and *grant left*$_2$ in the object net $phi_2$, i.e. the transition *object-object communication* in the system net fires and $phi_2$ gives up his/her left fork.

**Remark 6.1.6 (Elementary object systems and AHONI-systems)**
The translation of elementary object systems into AHONI-systems shown exemplary in Ex. 6.1.5 can be defined in a formal way but is beyond the scope of this thesis. Such a translation leads to equivalent systems wrt. their firing behavior if the net inscriptions in the elementary system are not restricted to specific object nets. Otherwise we have to extend our concept of AHONI-systems by some simple methods to guard the object nets, which should be moved along some transitions.

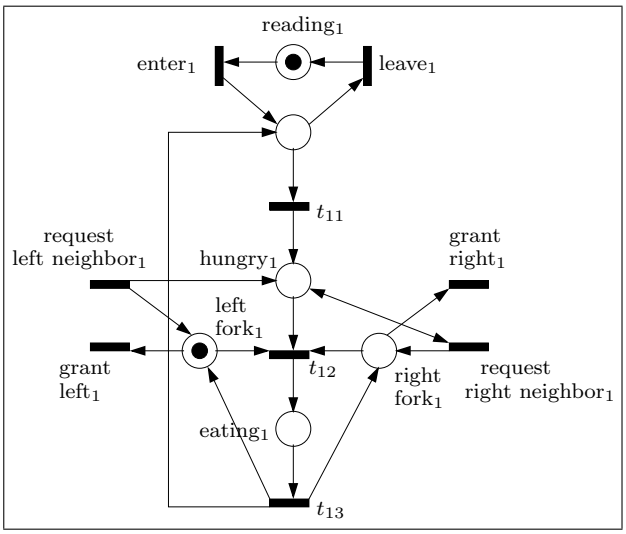



Figure 6.12: Token net of philosopher $phi_1$

## 6.2 AHO-Net and Rule Systems

In this section we propose the new paradigm "nets and rules as tokens", where in addition to nets as tokens also rules as tokens are considered. The rules can be used to change the structure of the token nets. This leads to the new concept of algebraic higher-order net and rule (AHONR)-systems, which allows the integration of the token game with rule-based transformations of P/T-systems. The new concept is based on AHO-nets with rule-based transformations. Here, we use the notion of token net instead of object net in order to avoid confusion with features of object-oriented modeling. Instead our intention is to consider the change of the net structure as rule-based transformations of Petri nets in the sense of graph transformation systems [Ehr79, Roz97]. Moreover, the interaction of the token game and transformation of nets - as considered in this thesis - has not been investigated up to now. This concept has interesting applications in all areas where dynamic changes of the net structure have to be considered while the system is still running. This applies especially to flexible workflow systems (see [vdA02]) and medical information systems (see [Hof00] and Chapter 8).

The new concept based on AHL-nets has already been published in [HEM05] and has been introduced with the case study "House of Philosophers", a dynamic extension of the well-known dining philosophers (see also Section 2.4). By contrast, in this section the new concept is based on AHO-nets. Since AHO-nets are grounded on higher-order algebras (see Section 3.1), we are able to give a set theoretic definition of domains and operations. In order to model nets and rules as tokens we present a specific higher-order signature together with a corresponding higher-order algebra with specific types for P/T-systems and rules. Moreover, there are operations corresponding to the firing of a transition and applying a rule to a P/T-system respectively. To obtain an algebraic specification as well we need algebraic higher-order specifications as presented in Chapter 7.

In order to allow P/T-systems and rules as tokens of an AHO-net we provide the higher-order signature, called NR-SIG, together with a suitable NR-SIG-algebra $A$, where NR refers to nets and rules.

**Definition 6.2.1 (NR-SIG Signature)**
The signature NR-SIG is given by
NR-SIG =

| | |
|---|---|
| sorts: | $Transitions, Places, System, Mor, Rules$ |
| opns: | $tt$, $ff$: $Pred(unit)$ |
| | $fire : (System \star Transitions \to System)$ |
| | $transform : (Rules \star Mor \to System)$ |
| | $cod : (Mor \to System)$ |
| | $eq : Pred(System \star System)$ |
| | $\& : Pred(Pred(unit) \star Pred(unit))$ |

and for each function type $(type_1 \to type_2) \in S^{\to}$ we have an application symbol $apply^{type_1,type_2} : (type_1 \to type_2) \star type_1 \rightharpoonup type_2$.

**Definition 6.2.2 (NR-SIG Algebra)**
Given vocabularies $T_0$ and $P_0$, the carrier of the NR-SIG-algebra $A$ for P/T-systems and rules as tokens is given by

- $A_{Transitions} = T_0$, $A_{Places} = P_0$,
- $A_{Pred(unit)} = \{true, false\}$,
- $A_{System}$ the set of all P/T-systems over $T_0$ and $P_0$, i.e.
  $A_{System} = \{PN | PN = (P, T, pre, post, M)$ P/T-system, $P \subseteq P_0, T \subseteq T_0\}$,
- $A_{Mor}$ the set of all P/T-system morphisms for $A_{System}$, i.e.
  $A_{Mor} = \{f | f : PN \longrightarrow PN'$ P/T-morphism with $PN, PN' \in A_{System}\}$,
- $A_{Rules}$ the set of all rules of P/T-systems, i.e.
  $A_{Rules} = \{r | r = (L \xleftarrow{i_1} I \xrightarrow{i_2} R)$ rule of P/T-systems with strict inclusions $i_1, i_2\}$,
- $A_{(System \star Transitions \to System)} = \{fire\}$,
- $A_{(Rules \star Mor \to System)} = \{transform\}$,
- $A_{(Mor \to System)} = \{cod\}$,
- $A_{Pred(System \star System)} = \{\stackrel{E}{=}\}$, and
- $A_{Pred(Pred(unit) \star Pred(unit))} = \{\wedge\}$

For all $type \in BFS^{\to}$ different from the types above we have

$$A_{type} = \emptyset.$$

The carrier is extended to the set of types of $S^{\to}$ by

$$A_{unit} := \{()\} \text{ and}$$
$$A_{type_1 \star \ldots \star type_n} := A_{type_1} \times \ldots \times A_{type_n}.$$

The partial operations of NR-SIG-algebra $A$ are defined by

- $tt_A = true$, $ff_A = false$, and
  $apply_A^{Pred(unit),unit} : \{true, false\} \times A_{unit} \multimap A_{unit}$
  with
  $$(x.())^A = \begin{cases} () & \text{if } x = true \\ undef & \text{if } x = false \end{cases}$$

- $\mathit{fire}_A = \mathit{fire}$ and
  $\mathit{apply}_A^{(System \star Transitions), System} : \{fire\} \times (A_{System} \times T_0) \multimap A_{System}$
  for $PN = (P, T, pre, post, M) \in A_{System}$ and $t \in T_0$ with

$$(fire.(PN, t))^A = \begin{cases} (P, T, pre, post, M \ominus pre(t) \oplus post(t)) & \text{if } t \in T, pre(t) \leq M \\ undef & \text{else} \end{cases}$$

- $transform_A(()) = transform$ and
  $\mathit{apply}_A^{(Rules \star Mor), System} : \{transform\} \times (A_{Rules} \times A_{Mor}) \multimap A_{System}$
  for $r \in A_{Rules}$ and $m \in A_{Mor}$ with

$$(transform.(r, m))^A = \begin{cases} H & \text{if } r \text{ is applicable at match } m \\ undef & \text{else} \end{cases}$$

  where for $L \xrightarrow{m} G$ and $r$ is applicable at match $m$ we have a direct transformation $G \overset{r}{\Longrightarrow} H$,

- $\mathit{cod}_A = \mathit{cod}$ and
  $\mathit{apply}_A^{Mor, System} : \{cod\} \times A_{Mor} \multimap A_{System}$
  for $m = (f : PN_1 \longrightarrow PN_2) \in A_{Mor}$ with

$$(cod.m)^A = PN_2,$$

- $eq_A(()) = \overset{E}{=}$ and
  $\mathit{apply}_A^{(System \star System), unit} : \{\overset{E}{=}\} \times (A_{System} \times A_{System}) \multimap A_{unit}$
  for $PN_1, PN_2 \in A_{System}$ with

$$(\overset{E}{=}.(PN_1, PN_2))^A = \begin{cases} () & \text{if } PN_1 = PN_2 \\ undef & \text{else} \end{cases}$$

  where the equality symbol $exp \overset{E}{=} exp'$ is used to state existential equality, i.e. both sides denote the same value,

- $\&_A = \wedge$ and
  $\mathit{apply}_A^{(Pred(unit) \star Pred(unit)), unit} : \{\wedge\} \times (A_{Pred(unit)} \times A_{(Pred(unit))}) \multimap A_{unit}$
  with

$$(\wedge.((), ()))^A = true$$

  where $\wedge$ is the well-known conjunction symbol.

**Definition 6.2.3 (Algebraic Higher-Order Net and Rule Systems)**
Given the signature NR-SIG and the NR-SIG-algebra $A$ as above, an algebraic higher-order net and rule system $AHONRS = ((N, A), INIT)$ consists of an AHO-net $(N, A)$ (see Def. 3.2.7) with $\Sigma = (\text{NR-SIG}, X)$ where $X$ are variables over NR-SIG, and an initial marking $INIT$ such that

1. all places $p \in P$ are either
   - system places i.e. $p \in P_{Sys} = \{p \in P | type(p) = System\}$ or
   - rule places i.e. $p \in P_{Rules} = \{p \in P | type(p) = Rules\}$,

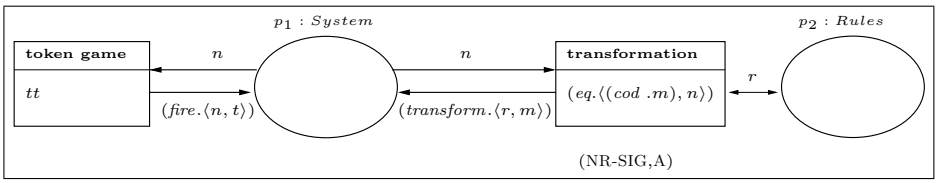



Figure 6.13: Basic higher-order net and rule system

2. all rule places $p \in P_{Rules}$ are contextual, i.e. for all transitions $t \in T$ connected with $p$ there exists a variable $r \in X$ such that $pre(t)_{|p} = post(t)_{|p} = r$, i.e. in the net structure of $(N, A)$ the connection between $p$ and $t$ is given by a double arrow labeled with the variable $r$.

**Remark 6.2.4 (Static Rules)**
Our notion of higher-order net and rule systems has static rules. This means that our rule tokens do not move and remain unchanged in the rule places (see Section 6.3 for extensions). According to our paradigm "nets and rules as tokens" we only allow system and rule places but no places which are typed by other types of NR-SIG.

**Remark 6.2.5 (Basic Higher-Order Net and Rule System)**
An interesting special case of AHONR-systems are basic AHONR-systems as presented in Fig. 6.13 with system place $p_1$ and rule place $p_2$, where the empty initial marking can be replaced by suitable P/T-systems resp. rules on these places. In general, a higher-order net and rule system with only one system place and one rule place is called basic higher-order net and rule system. Let us assume that the initial marking is given by a P/T-system $PN$ on place $p_1$ and a set $RULES$ of token rules on place $p_2$. Then $(PN, RULES)$ can be considered as reconfigurable P/T-system in the following sense: on the one hand we can apply the token game and on the other hand rule-based transformations of the net structure of $PN$. Moreover these activities can be interleaved. This allows to model changes of the net structure while the system is running. This is most important for changes on the fly of large systems, where it is important to keep the system running, while changes of the structure of the system have to be applied. It would be especially important to analyze under which conditions the token game activities are independent of the transformations. This problem is closely related to local Church-Rosser properties for graph resp. net transformations, which are valid in the case of parallel independence of transformations (see [Ehr79, EP04]).

**Example 6.2.6 (House of Philosophers)**
According to the requirements of the hurried philosophers in [SB01] the philosophers have the capability to introduce a new guest at the table, which - in the case of low level Petri nets - certainly changes the net structure of the token net representing the philosophers at the table.

In Section 2.4 we have given a solution of the case study "Hurried Philosophers" which is inspired by the case study of the hurried philosophers in [SB01]. As expected the system level of the "Hurried Philosophers" can also be modeled by using the concept of AHONR-systems. Especially for this example we have to extend the signature NR-SIG and the NR-SIG-algebra by two further constant symbols, i.e. the signature PHIL-SIG is given by

PHIL-SIG=NR-SIG +

opns: $coproduct : (System \times System \to System)$
$isomorphic : Pred(System \times System)$

where here and in other examples + stands for the disjoint union of sorts and operation symbols of two higher-order signatures. The PHIL-SIG-algebra $A'$ extends the carrier of the NR-SIG-algebra $A$ given above by

- $A'_{System \times System \to System} = \{coproduct\}$ and
- $A'_{Pred(System \times System)} = A_{Pred(System \times System)} \cup \{isomorphic\}$.

The partial operations of the PHIL-SIG-algebra $A'$ are the partial operations of the NR-SIG-algebra $A$ defined above and

- $coproduct_{A'} = coproduct$ and
  $apply_{A'}^{(System \star System), System} : \{coproduct\} \times (A_{System} \times A_{System}) \multimap A_{System}$
  for $PN_1, PN_2 \in A_{System}$ with
  $$(coproduct.(PN_1, PN_2))^{A'} = ((P_1 \uplus P_2), (T_1 \uplus T_2), pre_3, post_3, M_1 \oplus M_2)$$
  i.e. the disjoint union (the two P/T-systems are combined without interaction) and
- $isomorphic_A = isomorphic$ and
  $apply_{A'}^{(System \star System), unit} : \{\stackrel{E}{=}, isomorphic\} \times (A_{System} \times A_{System}) \multimap A_{unit}$
  for $PN_1, PN_2 \in A_{System}$ with
  $$(\stackrel{E}{=}.(PN_1, PN_2))^{A'} = (\stackrel{E}{=}.(PN_1, PN_2))^{A}$$
  and
  $$(isomorphic.(PN_1, PN_2))^{A'} = \begin{cases} () & \text{if } PN_1 \cong PN_2 \\ undef & \text{else} \end{cases}$$
  where $PN_1 \cong PN_2$ means that there is a strict P/T-morphism $f = (f_P, f_T)$ with $f : PN_1 \longrightarrow PN_2$ s.t. $f_P, f_T$ are bijective functions.

The system level of the "Hurried Philosophers" as an AHONR-system is depicted in Fig. 6.14, while the token level is given as in Section 2.4. Without going into detail, the algebraic high-level net in Fig. 2.3 and the AHONR-system in Fig. 6.14 are equivalent wrt. their operational behavior. This aspect will be discussed in Section 10.2.

In the solution of the case study using elementary object systems [Val01] each philosopher has his own place and the exchange of forks between the philosophers is realized by an interaction relation (see Section 6.1). By contrast in our case each table is modeled by its own P/T-system, which describes the states and the seating arrangement of present philosophers. In addition we use rule-based transformations to change the structure of P/T-systems, especially the states and the seating arrangement. In the sense of object-oriented modeling it could be considered splitting up the table with philosophers into a net table with only the table properties and nets for each philosopher at the table. In fact our approach allows to model such self-contained components but this would lead to a much more complex model. The advantage of our approach compared with elementary object systems is a more flexible modeling technique. While the AHO-net in Fig. 6.14 is fixed we can add further philosophers and philosophers at tables by adding further tokens of type *System* to our model. Analogously we can add further token rules to realize other kinds of transformations.

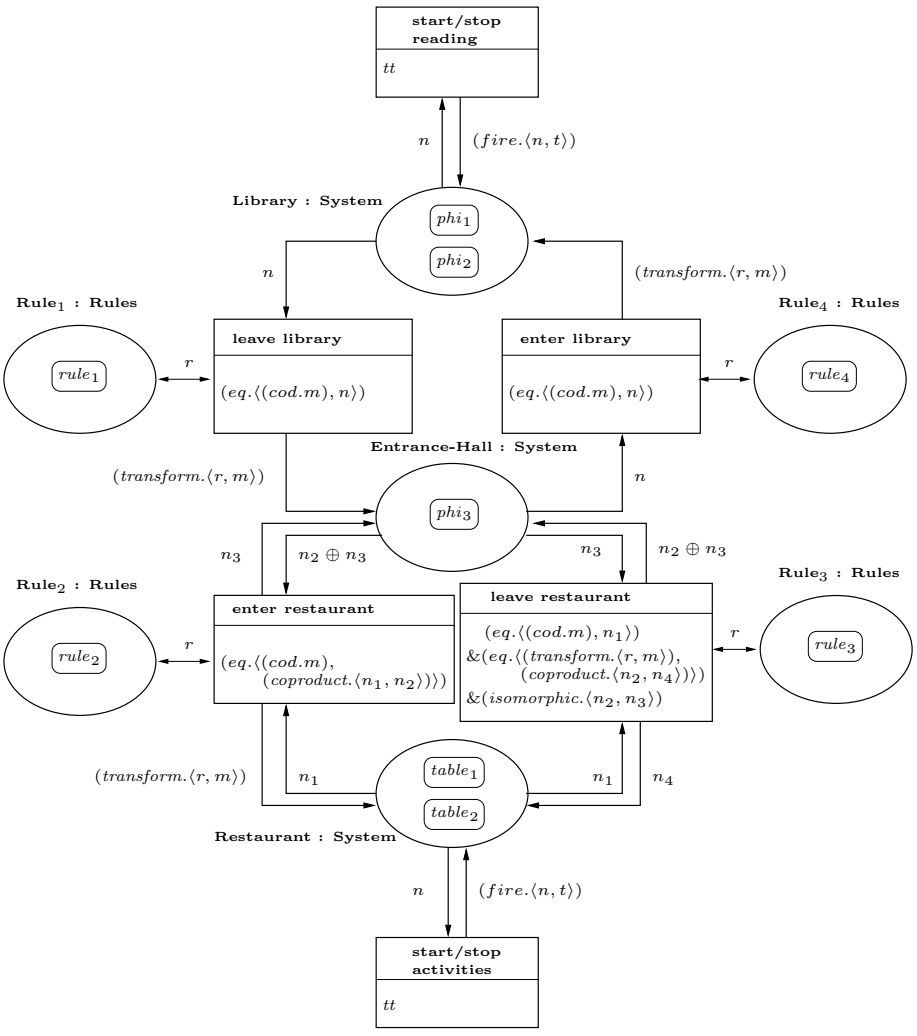



Figure 6.14: Algebraic higher-order net and rule system of "House of Philosophers"

The idea of controlled modification of token nets is discussed in the context of linear logic Petri nets [Far99] and feature structure nets [Wie01]. The difference to our approach is that in those approaches, the modification is not carried out by rule tokens but by transition guards. We are not restricted to define a specific token rule for each transition; instead we are able to give a (multi-)set of token rules as resources bound to each transition, which realize the local replacement of subnets.

In AHONR-systems the coupling of a set of rules as tokens to certain transitions is fixed due to the given net topology. In the subsequent section we will consider also the migration of rules as tokens. This means the mechanism of mobility and modification presented in our example could be transferred to rules as tokens in order to achieve even more expressive models. The mobility aspect of rules as tokens can be easily introduced by further transitions connecting places of the type *Rules*. However, the modification of rules as tokens (see [PP01]) requires an extension of the corresponding NR-SIG signature and NR-SIG-algebra.

An interesting aspect for future work is to study transformations of P/T-systems which preserve properties like safety or liveness. Especially in the area of workflow modeling the notion of soundness (which comprises liveness) is of importance (see e.g. [vdA98]). Here we can use the approach of property preserving rules (see [PU03] for an overview). To integrate these kinds of rules into AHONR-systems the set of rules $A_{Rules}$ of the NR-SIG-algebra $A$ would have to be restricted to property preserving rules.

## 6.3 AHO-Rule Systems

In Section 6.2 we have introduced the paradigm "nets and rules as tokens", where in addition to nets as tokens also rules as tokens are considered to change the net structure. In this section we consider rules as tokens leading to the concept of algebraic higher-order rule systems for mobile policies. The rules are used on the one hand for the specification of policy rules and on the other hand for the modification of policy rules, i.e. for the definition of new rules by reusing existing rules. So the higher-order rule system models distribution and modification of policy rules in a systematic and structured way. We give signature and corresponding algebra of rules and (local) transformations in the sense of the double-pushout approach and illustrate our concept by a small system inspired by the case study of a tax refund process [BFA99].

A policy is a set of rules which controls the behavior of complex systems. In [KPP02] a policy framework based on graph transformation is presented, which provides an intuitive visual formalism for the manipulation of policy-based systems. The policy framework is defined by a type graph and sets of policy rules, positive and negative constraints.

Mobile policies [CFJF02, DFJM00] are policies that can move along with the application or data that they refer to. Mobile policies can either supplement or override local policies, and can be used either to regulate access to the local resources (to protect the host) or to constraint use or access to the mobile code (to protect the guest). In a distributed environment applications migrate from site to site and the relative policy can migrate with the application it refers to, thus allowing each site to avoid locally storing policies for all possible applications. A framework in which mobile policies are attached to the relative application also facilitates the development of new applications and places the responsibility for the application-specific policy on the application designer.

Mobile policies may also need to be modified in prearranged ways to adapt to external requirements of specific domains. For example, certain local laws may require or forbid certain behavior of all applications executing locally (restrictions

on the use of encryption or on the length of the key in a cryptosystems, additional requirements to protect privacy) and therefore the constraints imposed by the policy may need to be adapted in moving from site to site.

We investigate how the distribution, the migration, and the modification of mobile policies can be modeled by using algebraic higher order rule (AHOR) systems which are based on AHO-nets. Due to the higher-order features, graphs and rules are allowed to be dynamic objects in AHOR-systems and the behavior of an AHOR-system simulates the modification needed to achieve the flexibility of adapting objects.

For our purpose, the AHOR-system gives an overview of the different locations where the mobile policies could reside. Furthermore, the coupling of a set of rules, which are used to modify policy rules with certain locations, which have the authority to modify the policy rules, is given by the net topology. The behavior of an AHOR-system simulates the application of a rule to a policy rule and describes the modification of the policy rule in order to achieve a more appropriate one. In this section policy rules are used for the specification of access control [San98]. Apart from this, the concept has interesting applications in all areas where individual rules are modified while the system is running.

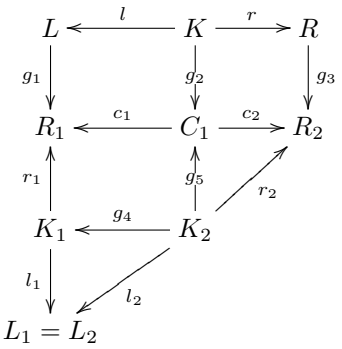
Rules and transformations are formalized on a rigorous mathematical foundation in the context of high-level replacement systems [EHKP91], a categorical generalization of the concept of graph transformation systems to other kinds of structures based on the double-pushout approach. A high-level replacement system is defined by an arbitrary category and a distinguished class of morphisms used to form rules, i.e. rules in the double-pushout approach are given as a span of two morphisms, and its application is achieved by two pushouts. Think of these rules as replacement systems, where the left-hand side of the rule is replaced by the right-hand side. Moreover, we reuse policy rules, i.e. we modify policy rules in the sense of inheritance [PP01].

Technically, we give a signature and corresponding algebra of rule-based modifications and specific operations for the modification of mobile policies. Here we use the approach given in [PP01] to obtain suitable operations for the modification of rules. We use this signature and algebra as the data type part of AHOR-systems to denote the application of rules in the net inscriptions. Then the behavior of the AHOR-system simulates the modification of rules to obtain a new and more appropriate policy rule.

The advantage of our approach is twofold. On the one hand the AHOR-system manages the distribution of rules in a systematic and structured way. Large sets of rules are divided into smaller ones, which are locally bound to some transitions. On the other hand AHOR-systems are flexible in respect of the replacement of rules. Formally we exchange rules by exchanging the corresponding tokens to realize other kinds of transformations, while the system net is fixed.

To define new rules by reusing existing rules we use the approach given in [PP01], where different concepts are presented for the modification of one rule by another one. In this section we describe the more general form of inheritance in more detail. Inheritance is meant in the sense that existing rules are reused by extending them to provide the desired behavior, i.e. the new rule coincides with the left hand side of the old rule, but has a different right hand side.

**Definition 6.3.1 (Inheritance)**
Let a rule $p_1 = (L_1 \xleftarrow{l_1} K_1 \xrightarrow{r_1} R_1)$ be given. Then the modification of the right-hand side $R_1$ via a rule $q = (L \xleftarrow{l} K \xrightarrow{r} R)$ is illustrated in the following diagram,




where

- $R_2$ is the object resulting from the direct transformation via $q$ of $R_1$,
- $K_2$ is the common part of $C_1$ and $K_1$, and
- $L_2$ is just the unchanged left-hand side of $p_1$.

Then the new rule $p_2$ is given by $p_2 = (L_2 \xleftarrow{l_2} K_2 \xrightarrow{r_2} R_2)$ with $l_2 = l_1 \circ g_4$ and $r_2 = c_2 \circ g_5$.

**Example 6.3.2 (Inheritance)**
An example of the modification of one rule by another rule using the concept of inheritance is depicted in Fig. 6.23. Thus, we have a specific operation $inheritance_{\mathcal{A}} : \mathcal{A}_{Rules} \times \mathcal{A}_{Rules} \times \mathcal{A}_{Mor} \to \mathcal{A}_{Rules}$, so that $inheritance_A(p_2, p_{14}, g_1) = p_{11}$, where $p_2$ is the policy rule of company $C1$ (see Fig. 6.18), $p_{14}$ is a rule for the modification of $p_2$ (see Fig. 6.21), $g_1$ is a suitable occurrence morphism, and $p_{11}$ is the policy rule of company $C2$ (see Fig. 6.23).

The category of (labeled) directed graphs with the distinguished class of injective, colour preserving graph morphisms has been checked to be a high-level replacement category (see [EHKP91]) with the capacity to guarantee Church-Rosser and Parallelism Theorems for high-level replacement. For a detailed definition of (labeled) graphs, graph morphisms, graph rules and direct transformation we refer to [Roz97]. An example of a direct transformation can be found in Fig. 6.23 where the rule $p_{14}$ (see Fig. 6.21) is applied to the object $R_2$.

In order to allow P/T-systems and rules as tokens of an AHO-net we provide a specific higher-order signature, called MOBILE POLICIES, together with a suitable MOBILE POLICIES-algebra $A$.

**Definition 6.3.3 (Mobile Policies-Signature)**
The signature MOBILE POLICIES is given by

MOBILE POLICIES =
sorts: $Labels, Edges, Nodes, Graph, Mor, Rules, SetRules$
opns: $tt$, $ff$: $Pred(unit)$
$inherit : (Rules \star Rules \star Mor \to Graph)$
$right : (Rules \to Graph)$
$cod : (Mor \to Graph)$
$\{\ \} : (Rules \to SetRules)$
$\in$: $Pred(Rules \star SetRules)$
$\cup, \setminus : (SetRules \star SetRules \to SetRules)$
$eq_{Graph} : Pred(Graph \star Graph)$
$\& : Pred(Pred(unit) \star Pred(unit))$
and for each function type $(type_1 \to type_2) \in S^{\to}$ we have an application symbol $apply^{type_1, type_2} : (type_1 \to type_2) \star type_1 \rightharpoonup type_2$.

**Definition 6.3.4 (Mobile Policies-Algebra)**
Given vocabularies $L_0, E_0$, and $V_0$, the carrier of the MOBILE POLICIES-algebra $A$ for graphs rules as tokens is given by

- $A_{Label} = L_0, A_{Edges} = E_0$, and $A_{Nodes} = V_0$,
- $A_{Pred(unit)} = \{true, false\}$,
- $A_{Graph}$ the set of all (labeled) graphs over $L_0, E_0$, and $V_0$, i.e.
  $A_{Graph} = \{G | G = (V, E, source, target, nlabel, elabel)$ (labeled) graph,
  $V \subseteq V_0, E \subseteq E_0\}$,
- $A_{Mor}$ the set of all graph morphisms for $A_{Graph}$, i.e.
  $A_{Mor} = \{f | f : G \to G'$ graph morphism with $G, G' \in A_{Graph}\}$,
- $A_{Rules}$ the set of all graph rules, i.e.
  $A_{Rules} = \{p | p = (L \xleftarrow{l} K \xrightarrow{r} R)$ graph rule with inclusions $l, r\}$,
- $A_{SetRules}$ the finite power set of the set of graph rules, i.e.
  $A_{SetRules} = \{R | R \in \mathcal{P}_{fin}(A_{Rules})\}$,
- $A_{(Rules \star Rules \star Mor \to Graph)} = \{inherit\}$,
  $A_{(Rules \to Graph)} = \{right\}$, and
  $A_{(Mor \to Graph)} = \{cod\}$,
- $A_{(Rules \to SetRules)} = \{\{\ \}\}$,
  $A_{Pred(Rules \star SetRules)} = \{\in\}$, and
  $A_{(SetRules \star SetRules \to SetRules)} = \{\cup, \setminus\}$,
- $A_{Pred(Graph \star Graph)} = \{\overset{E}{=}\}$ and
  $A_{Pred(Pred(unit) \star Pred(unit))} = \{\wedge\}$.

For all $type \in BFS^{\to}$ different from the types above we have

$$A_{type} = \emptyset.$$

The carrier is extended to the set of types of $S^{\to}$ by

$$A_{unit} := \{()\} \text{ and}$$
$$A_{type_1 \star \ldots \star type_n} := A_{type_1} \times \ldots \times A_{type_n}.$$

The partial operations of MOBILE POLICIES-algebra $A$ are defined by

- $tt_A = true$, $ff_A = false$, and
  $apply_A^{Pred(unit), unit} : \{true, false\} \times A_{unit} \multimap A_{unit}$
  with
  $$(x.())^A = \begin{cases} () & \text{if } x = true \\ undef & \text{if } x = false \end{cases}$$
- $inherit_A = inherit$ and
  $apply_A^{(Rules \star Rules \star Mor), Rules} : \{inherit\} \times (A_{Rules} \times A_{Rules} \times A_{Mor}) \multimap A_{Rules}$

for $p_1, q \in A_{Rules}$ with $p_1 = (L_1 \xleftarrow{l_1} K_1 \xrightarrow{r_1} R_1)$ and $q = (L \xleftarrow{l} K \xrightarrow{r} R)$, and $g \in A_{Mor}$ with

$$(inherit.(p_1, q, g))^A = \begin{cases} p_2 & \text{if } q \text{ is applicable at match } g \\ undef & \text{else} \end{cases}$$

where for $L \xrightarrow{g} R_1$ and $q$ is applicable at match $g$ we get a new rule $p_2$ given by $p_2 = (L_2 \xleftarrow{l_2} K_2 \xrightarrow{r_2} R_2)$ (see Def. 6.3.1),

- $right_A = right$ and
  $apply_A^{Rules,Graph} : \{right\} \times A_{Rules} \multimap A_{Graph}$
  for $r \in A_{Rules}$ and $p = (L \xleftarrow{l} K \xrightarrow{r} R)$ with
  $$(right.r)^A = R$$

- $cod_A = cod$ and
  $apply_A^{Mor,Graph} : \{cod\} \times A_{Mor} \multimap A_{Graph}$
  for $m = (f : G_1 \longrightarrow G_2) \in A_{Mor}$ with
  $$(cod.m)^A = G_2,$$

- $\{\ \}_A = \{\ \}$, $\in_A = \in$, $\cup_A = \cup$, and $\setminus_A = \setminus$ with the usual set-theoretic interpretations,

- $eq_{Graph,A} = \stackrel{E}{=}$ and
  $apply_A^{(Graph \star Graph),unit} : \{\stackrel{E}{=}\} \times (A_{Graph} \times A_{Graph}) \multimap A_{unit}$
  for $G_1, G_2 \in A_{Graph}$ with
  $$(\stackrel{E}{=}, (G_1, G_2))^A = \begin{cases} () & \text{if } G_1 = G_2 \\ undef & \text{else} \end{cases}$$
  where the equality symbol $exp \stackrel{E}{=} exp'$ is used to state existential equality, i.e. both sides denote the same value,

- $\&_A = \wedge$ and
  $apply_A^{(Pred(unit) \star Pred(unit)),unit} : \{\wedge\} \times (A_{Pred(unit)} \times A_{(Pred(unit))}) \multimap A_{unit}$
  with
  $$(\wedge.((), ()))^A = true$$
  where $\wedge$ is the well-known conjunction symbol.

**Definition 6.3.5 (Algebraic Higher-Order Rule Systems)**
Given the signature MOBILE POLICIES and the MOBILE POLICIES-algebra $A$ as above, an algebraic higher-order rule system $AHORS = ((N, A), INIT)$ consists of an AHO-net $(N, A)$ (see Def. 3.2.7) with $\Sigma = (\text{MOBILE POLICIES}, X)$ where $X$ are variables over P/T-SYSTEM, and initial marking $INIT$ of $(A, N)$ such that

1. all places $p \in P$ are either
   - rule places i.e. $p \in P_{Rules} = \{p \in P | type(p) = Rules\}$ or
   - policy places i.e. $p \in P_{Policies} = \{p \in P | type(p) = SetRules\}$,

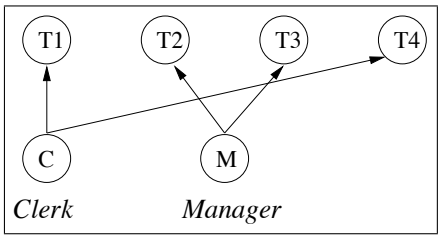



Figure 6.15: Role authorization

2. all rule places $p \in P_{Rules}$ are contextual, i.e. for all transitions $t \in T$ connected with $p$ there exists a variable $q \in X$ such that $pre(t)_{|p} = post(t)_{|p} = q$, i.e. in the net structure of $(N, A)$ the connection between $p$ and $t$ is given by a double arrow labeled with the variable $q$.

**Example 6.3.6 (Tax Refund Process)**
In order to illustrate the concepts described above we present a small system inspired by the case study of a tax refund process given in [BFA99]. The main idea of our example is to model mobile policies which move around between different companies. The policy rules are not fixed once and for all because each company expects specific policy rules. Our example is restricted in the sense that we do not take into account all aspects of the policy framework presented in [KPP02]. In this example we specify policy rules, which are building the accepted system states, and assume that the policy rules are built over a given type graph. Furthermore, we do not focus on the application of policy rules to the actual state of an object.

The example deals with a tax refund process which is a simplified version of the workflow introduced in [BFA99]. The workflow representing the tax refund process in company $C1$ consists of four tasks to be executed sequentially:

- Task $T1$: A clerk prepares a check for a tax refund.
- Task $T2$: A manager can approve or disapprove the check. This task must be performed by two managers.
- Task $T3$: The decisions of the managers are collected and the final decision is made by a manager. Her/his decision is a consequence of the outcome of task $T2$, i.e. (s)he does not decide about the tax refund.
- Task $T4$: A clerk issues if both managers approved or voids if one manager disapproved the check on the result of task $T3$.

By contrast, the tax refund process in the company $C2$ is altered in task $T2$ and task $T4$, while Task $T1$ and Task $T3$ are left unchanged:

- Task $T2$: A manager can approve or disapprove the check. This task must be performed by one manager.
- Task $T4$: A clerk issues if the manager approved or voids if the manager disapproved the check on the result of task $T3$.

In Fig. 6.15 each task is related to a role which can execute the task, e.g. the role *Clerk* can execute task $T1$ and task $T4$.

Our first goal is a representation of the system level as an AHOR-system so that the system shows on the one hand the distribution of policy rules and on the other hand the coupling of rules to certain transitions. Thus, the firing behavior of the AHOR-system describes the migration and local transformation of policy rules.

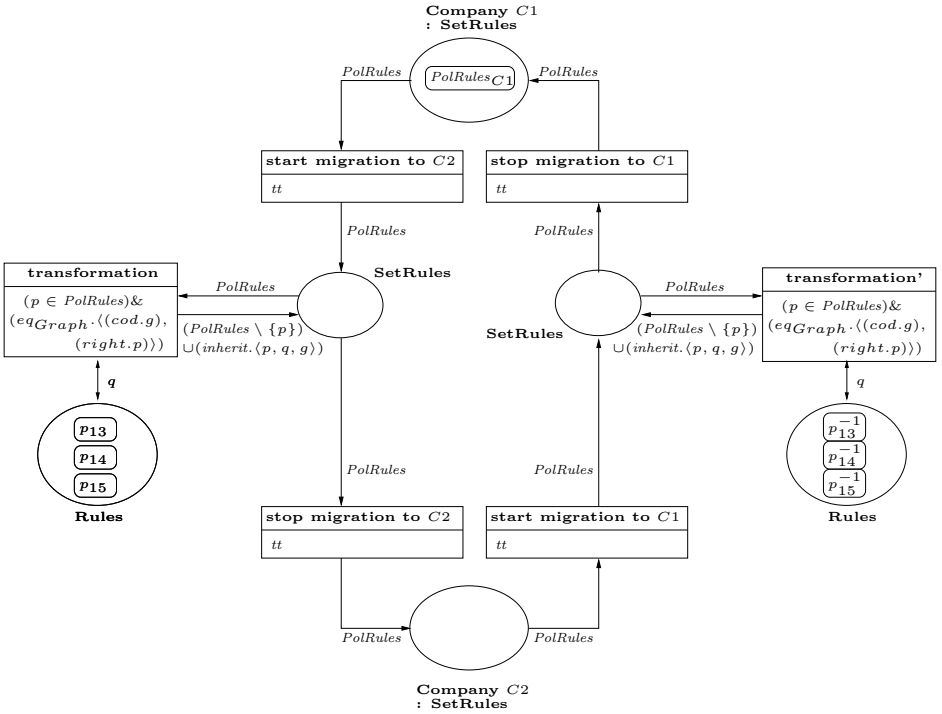



Figure 6.16: Algebraic higher-order rule system "Tax Refund Process"

In Fig. 6.16 we sketch a solution for the example of the tax refund process. The initial marking and the net inscriptions of the AHOR-system in Fig. 6.16 are built over the signature Mobile Policies and the corresponding algebra $A$. There are four different locations where policy rules can stay: the company $C1$, the company $C2$, and during the migration processes between $C1$ and $C2$, or $C2$ and $C1$. Each location becomes represented by its own place in the AHOR-system in Fig. 6.16. The initial marking consists of the policy rules $PolRules_{C1}$ of company $C1$ and specific rules for the modification of policy rules.

Policy rules may move around, which means they might leave and enter the company $C1$ and they might leave and enter the company $C2$. The mobility aspect of the policy rules is modeled by transitions termed in an obvious way in our system net in Fig. 6.16. While policy rules are moving around they have to be changed in a certain way using the concept of inheritance. For this reason there are other kinds of rules, $p_{13} - p_{15}$ and $p_{13}^{-1} - p_{15}^{-1}$, to guarantee the correct modification of policy rules. Here these rules are used as resources bound to corresponding transitions.

Thus the object level consists of two different kinds of objects: policy rules and rules for the modification of policy rules. In the following we will explain the set of policy rules $PolRules_{C1}$ of company $C1$ in more detail. Although they are based on the rules given in [KPP02], we use the double-pushout approach in this section instead of the single-pushout approach and we have to take care of the so called gluing condition. For simplicity reasons we avoid negative application conditions, i.e. we cannot distinguish between the users which are involved in the tax refund process.

The set of policy rules of company $C1$ is given by $PolRules_{C1} = \{p_1, \ldots, p_9\}$. The rule $p_1$ (see Fig. 6.17) creates a new check by a user associated to the role *Clerk*. A user is represented by a node of type $U$. The two loops of the *check* node

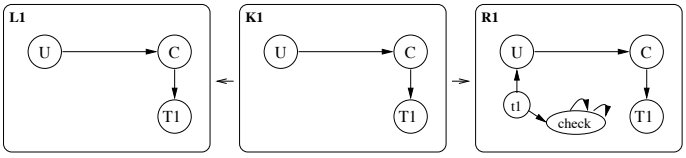



Figure 6.17: Rule $p_1$ for task $T1$ in company $C1$

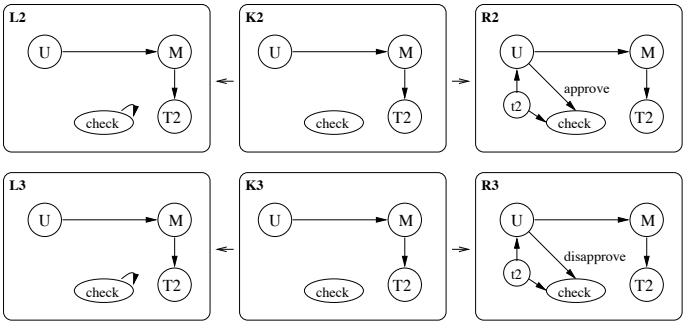



Figure 6.18: Rules $p_2$ and $p_3$ for task $T2$ in company $C1$

indicate that the recommendation for a tax refund has to be performed by two managers. The rules $p_2$ and $p_3$ realize the process of task $T2$ (see Fig. 6.18), i.e. a manager approves or disapproves the check. In detail, a loop from the *check* node is deleted and an edge between the *user* node and the *check* node labeled with the recommendation is created. Thus, these rules are only applicable if there is a loop attached to the *check* node. The rules $p_4$, $p_5$ and $p_6$ realize the process of task $T3$ (see Fig. 6.19), i.e. the collection of the decisions. The two edges which model the recommendations of the two managers are deleted and a loop of the *check* node is created labeled by *issue* if both managers approve and by *void* if one of the managers disapproves. In all three cases the manager does not decide directly because the decision is based on the previous recommendations. The rules $p_7$ and $p_8$ realize the process of task $T4$ (see Fig. 6.20), i.e. a clerk issues or voids the check. The end of the workflow for this check is indicated by changing the color of the *check* node and deleting the corresponding loop. Finally, the tax refund process is finished by using rule $p_9$ (see Fig. 6.20) to delete the *check* node and all connected nodes t1-t4 and adjacent edges.

The set of policy rules described above may move around between the two companies. Because each company expects specific policy rules, some rules have to be modified during the migration. In detail the following policy rules of company $C1$ have to be modified to respect the requirements of company $C2$:

- preparation of the check (see rule $p_1$ in Fig. 6.17)
- approval of the check (see rule $p_2$ in Fig. 6.18)
- disapproval of the check (see rule $p_3$ in Fig. 6.18)

To integrate the modification of policy rules into our model, we need an operation to achieve new rules by reusing existing ones. Here we use the approach of local transformation as presented in [PP01] to get a new rule which coincides with the left hand side of the "old" one but which has a different right hand side and (in general) a different interface part.

The modification of the rule $p_1$ (see Fig. 6.17) is attained via the rule $p_{13}$ (see Fig. 6.21). Here we use the transition *transformation* of the AHO-net in Fig. 6.16.

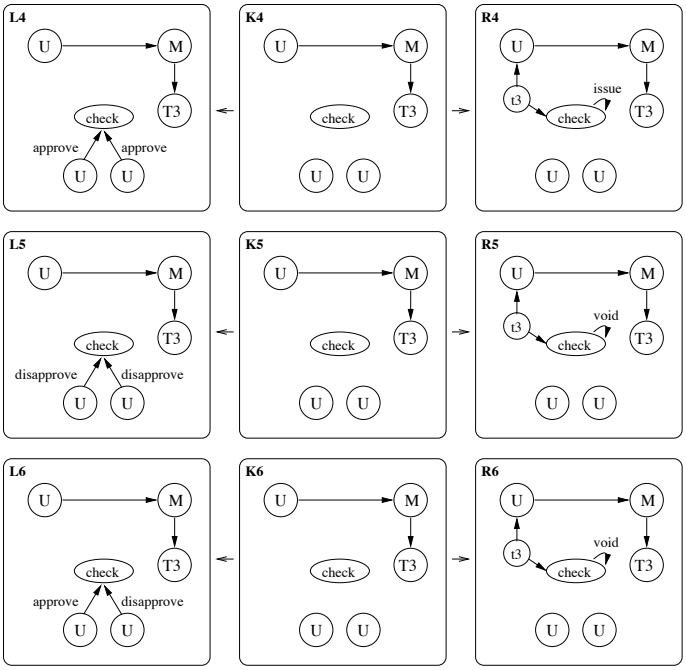



Figure 6.19: Rules $p_4$, $p_5$ and $p_6$ for task $T3$ in company $C1$

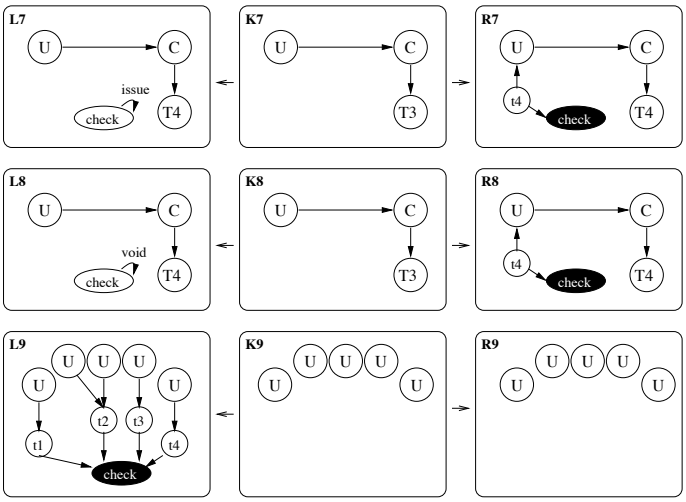



Figure 6.20: Rules $p_7$, $p_8$ and $p_9$ for task $T4$ in company $C1$

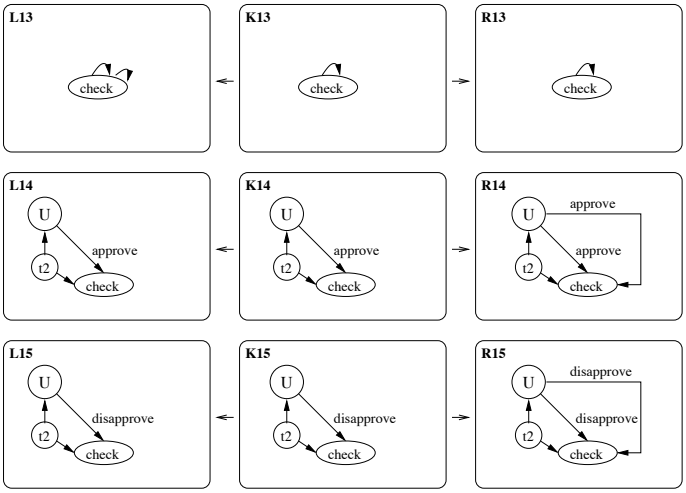



Figure 6.21: Rules $p_{13}$, $p_{14}$ and $p_{15}$ for the modification of policy rules

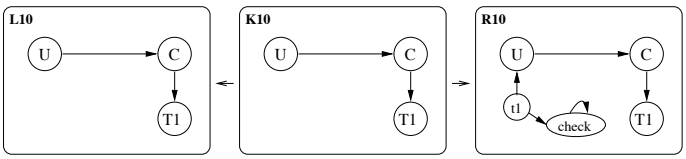



Figure 6.22: Rule $p_{10}$ for task $T1$ in company $C2$

First, the net inscriptions in the environment of the transition are evaluated, i.e. the variable *PolRules* is assigned to the set of policy rules $PolRules_{C1}$, the variable $p$ to the rule $p_1$, the variable $q$ to the rule $p_{13}$, and the variable $g$ to an occurrence morphism $g_1 : L \longrightarrow G$. The firing condition $cod\ g = cod\ r$ requires $L = L_{13}$ and $G = R_1$. The transition *transformation* is enabled under this assignment, i.e. the evaluation of the net inscriptions is defined. Then the evaluation of the term $inherit(p, q, g)$ computes the modification of rule $p_1$ via the rule $p_{13}$ using the concept of inheritance. We obtain the new policy rule $p_{10}$ of company $C2$ depicted in Fig. 6.22. The rule $p_{10}$ adds one loop to the *check* node because one manager has to approve or disapprove the check.

For the modification of the rule $p_2$, resulting in the new rule $p_{11}$ (see Fig. 6.23), we use a different variable assignment, i.e. the variable $p$ is assigned to the rule $p_2$, the variable $q$ to the rule $p_{14}$, and the variable $g$ to an occurrence morphism $g_1 : L \longrightarrow G$ so that $L = L_{14}$ and $G = R_2$. Then $R_{11}$ is the object resulting from the direct transformation via $p_{14}$ of $R_2$, $K_{11}$ is the common part of $C_1$ and $K_2$, and $L_{11}$ is just the unchanged left hand side of $p_2$ (see Fig. 6.23). The new rule $p_{11} = (L_{11} \leftarrow K_{11} \rightarrow R_{11})$ adds two edges with the same label *approve* to the *check* node because the clerk issues if one manager has approved. Analogously, the modification of the rule $p_3$ (see Fig. 6.18) via the rule $p_{15}$ (see Fig. 6.21) results in the policy rule $p_{12}$ of company $C_2$ (see Fig. 6.24). Finally, the set of policy rules of company $C2$ consists of the rules $p_4$ - $p_{12}$ (see Figs. 6.19, 6.20, 6.22, 6.23 and 6.24), where the three rules concerning the preparation of a check and the approval or disapproval of a check are modified.

Summarizing, we have presented a powerful technique to model mobile policies using AHOR-systems in order to achieve highly expressive models. We have re-

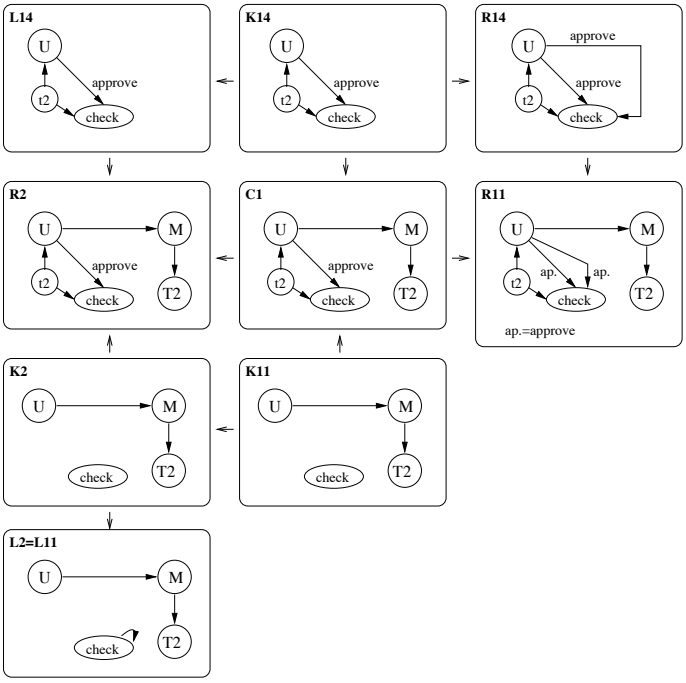



Figure 6.23: Modification of $p_2$ via $p_{14}$ to achieve $p_{11}$

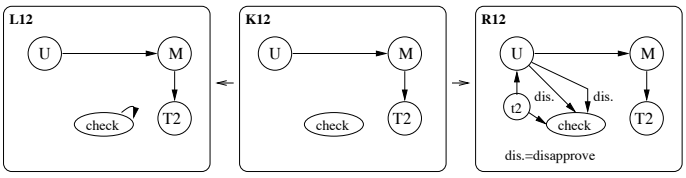



Figure 6.24: Rule $p_{12}$ for task $T2$ in company $C2$

viewed the concept of graph rules and the concept of local transformations and have transferred these concepts into a MOBILE POLICIES-signature and a MOBILE POLICIES-algebra. Afterwards we have explained the structure and behavior of AHOR-systems. We have illustrated the use of AHOR-systems for mobile policies through the example of the tax refund process. Our system describes the migration of policy rules from one company $C1$ to another company $C2$. Moreover, policy rules become modified during the migration process by specific rules, so that the application of these rules results in new rules matching the requirements of the companies. Thus, local transformations become effectively included into the system enabling the system to transform rules in a formal way.

The main advantage of using AHOR-systems is their flexibility in respect of introducing new rules to the system. While the system level is fixed, we can add further policy rules and rules for the modification of policy rules by adding further tokens of type *Rules* to our model. Note that the structure of these rules can be different from the structure of the rules presented in Example 6.3.6.

An interesting aspect of future work is to integrate not only the specification of policy rules into our system but also all other aspects of the policy framework presented in [KPP02], i.e. the type graph, the set of positive and negative graphical constraints and the application of policy rules to build the actual system state.

In Section 6.2 we have presented algebraic higher-order net and rule systems, where the application of rules to P/T-nets is modeled by corresponding operations. Thus, the system presented in this example can be extended by these concepts, i.e. the set of rules can be extended to a graph grammar.

In this section we have used the concept of inheritance to modify policy rules. But there are other concepts, e.g. the concept of specialization, where properties are added to policy rules or the concept of analogy, where a policy rule becomes reused in a different context. The approach given in [PP01] is a good starting point to obtain a suitable specification for this.

# Chapter 7

# Specifications and Implementation Aspects

In the previous chapter we have presented an explicit and set theoretical version of different kinds of higher-order algebras which defines different classes of AHO-nets for specific application domains. But it is also interesting to present higher-order specifications of these higher-order algebras. Unfortunately first-order algebraic specifications in the sense of [EM85] or CASL (Common Algebraic Specification Language) [Mos04] are not suitable for this purpose. So we actually need higher-order features which can be provided for instance by higher-order specifications presented in [Wol05], an extension of higher-order signatures defined in Section 3.1 by conditional existence equations, or by HASCASL [SM02, Mos05], a higher-order extension of CASL.

In this thesis we have introduced AHO-nets where for various reasons the data type part is given by higher-order signatures instead of higher-order specifications. The main reason is a theoretical one. Using higher-order specifications makes the theory more complicated because we have to take axioms into account. Here we are forced to decide which kind of higher-order specification morphisms are suitable for our purpose. For instance the axioms should be preserved in the sense of translated axioms being a subset of the axioms in the codomain specification, or the translated axioms should be derivable from the axioms in the codomain specification, i.e. axioms are transformed into theorems. Although the theory presented in this thesis would not have been affected by axioms - it is a well-known fact that the cocompleteness of the category of signatures implies the cocompleteness of the category of specifications [GB84]) - it is more likely for further research to get some results for AHO-nets with higher-order signatures, especially for rule-based transformations (see Section 10.1). A more general reason for using higher-order signatures instead of higher-order specifications in our basic notion of AHO-nets is that the construction of the initial model is obviously much more complicated in the partial and higher-order case than in the total and classical case. We are convinced that the explicit and set theoretical version of different kinds of higher-order partial algebras as presented in this thesis conveys our concepts in a much better way. Nevertheless, extending our approach by conditional existence equations enables us to use the term generated model instead of an explicit definition of an algebra. Moreover, it is a first step towards an implementation of our approach, because especially for HASCASL tools are available. Exemplarily some specifications are given in the next sections.

## 7.1 Higher-Order Specifications

Based on the well-developed theory of first-order partial algebras with conditional existence equations [Rei87, Wol91, CGRW95], the approach of higher-order partial algebras (see Section 3.1) is extended by conditional existence equations in [Wol05], enabling a direct link to approaches for implementing functional languages. This approach ensures the necessary semantical properties, i.e. the existence of higher-order partial algebras, freely generated by a set of variables and a set of existence equations, and the existence of free functor semantics. In the following we review the concepts of conditional existence equations and initial semantics for higher-order partial algebras as introduced in [Wol05]. Note that the definition of higher-order terms in Section 3.1 can be extended to higher-order (total) term algebras. But being total, the term algebra has not the nice property of being initial in the category of higher-order partial algebras. For the construction of this algebra we have to transform the syntactical tuples into semantical tuples by composition and decomposition functions, because the codomain of operations in higher-order signatures can consist of product types and the set of terms of type *unit* can consist of more than one term. Thus, we need a normalization of terms, such that there is a total and surjective evaluation from the set of terms into the corresponding higher-order term algebra, e.g. all terms of type *unit* have to be evaluated to the term $\langle\rangle$.

An existence equation on a set of variables is an expression $term_1 \stackrel{E}{=} term_2$ to be read $term_1$ and $term_2$ are defined and equal. Equations $term \stackrel{E}{=} term$ are abbreviated as $def\ term$ and called definedness judgments. A conditional existence equation is a sentence of the form $(X : G \Rightarrow term_1 \stackrel{E}{=} term_2)$, where $G$ is a set of existence equations on $X$, and $term_1 \stackrel{E}{=} term_2$ is an existence equation on $X$, to be read $term_1 \stackrel{E}{=} term_2$ holds on the domain of $G$. The satisfaction of a sentence in a higher-order partial algebra is determined as usual, by holding of its atomic formulae wrt. variable valuation of (defined) values to all the variables that occur in them. Note that the value of a term wrt. a valuation may be undefined due to the partiality of the term evaluation function (see Section 3.1).

Let us look at the example of natural numbers presented in Section 3.3. To give a specification we extend the (main part) of the higher-order signature HO-NAT by the usual constructors *zero* and *succ* for natural numbers. The specification HO-NAT-SP would be as follows.

HO-NAT-SP =

| | |
|---|---|
| sorts: | $Nat$ |
| opns: | $zero : Nat$ |
| | $succ : (Nat \to Nat)$ |
| | $geq : Pred(Nat \star Nat)$ |
| | $sub : (Nat \star Nat \to Nat)$ |
| vars: | $x, y : Nat$ |
| axioms: | $def\ zero$ |
| | $def\ succ$ |
| | $def\ geq$ |
| | $def\ sub$ |
| | $def\ (succ.x)$ |
| | $(geq.\langle x, zero\rangle) = \langle\rangle$ |
| | $def\ (geq.\langle x, y\rangle) \Rightarrow (geq.\langle(succ.x), (succ.y)\rangle) = (geq.\langle x, y\rangle)$ |
| | $(sub.\langle x, zero\rangle) = x$ |
| | $def\ (geq.\langle x, y\rangle) \Rightarrow (sub.\langle(succ.x), (succ.y)\rangle) = (sub.\langle x, y\rangle)$ |

Recall that for instance $succ : (Nat \to Nat)$ is an abbreviation for $succ : unit \rightharpoonup (Nat \to Nat)$ and $succ$ is rather a (partial) constant symbol of function type $(Nat \to Nat)$ than an operation symbol between the basic type of natural numbers. Furthermore we omit application symbols and the application of constant symbols to terms of type $unit$ (e.g. $def\ succ$ is an abbreviation for $apply^{Nat \to Nat, unit}(succ, \langle\rangle) = apply^{Nat \to Nat, unit}(succ, \langle\rangle)$). The first four axioms guarantee that the constant symbols are total constants, i.e. they have to be defined in a corresponding model, while the fifth axiom ensures that the "successor" function becomes total as well. The last four axioms specify the expected behavior of the functions "greater or equal" and "subtraction" for natural numbers.

For the construction of the initial algebra wrt. the specification HO-NAT-SP we first restrict the set of terms to a subset consisting exactly of those terms, which are defined in each higher-order partial HO-NAT-SP-algebra. In a second step we construct the congruence relation which is induced by the term evaluation into the higher-order partial HO-NAT-SP-algebras. In detail, we use the intersection of these term evaluations. Finally, we apply the normalization function described above to guarantee that we really get a HO-NAT-SP-algebra. The initial algebra is then given by the quotient term algebra wrt. the restricted set of normalized terms and the congruence relation of normalized kernels. For a precise definition we refer to [Wol05]. Let $T$(HO-NAT-SP) denote the initial algebra of the specification HO-NAT-SP given above. Then we have for instance $T(\text{HO-NAT-SP})_{unit} = \{[\langle\rangle]\}$ and $T(\text{HO-NAT-SP})_{(Pred(nat \to nat))} = \{[geq]\}$.

Summarizing, we expect that our concept of AHO-nets could be extended so that the data type part takes higher-order specifications into account. But as mentioned in the introduction of this chapter, suitable specification morphisms have to be well-chosen and we feel that deeper analysis is necessary before these ideas can be put into practice.

## 7.2 Basic Concepts of HasCasl

The specification language Casl (Common Algebraic Specification Language) (see e.g. [Mos04]) has been designed by CoFI, the Common Framework Initiative for algebraic specification and development. It is an expressive language for specifying requirements and design for conventional software. From Casl simpler languages are obtained by restrictions, e.g. the basic specifications in Casl allow declaration of sorts, operations (both total and partial), predicates and the use of formulae of first-order logic for stating axioms. More advanced features include subsorting, structures and architectural specification.

HasCasl has been introduced in [SM02] as a higher-order extension of Casl. HasCasl combines the simplicity of algebraic specifications and higher-order features towards a specification of functional programs, in particular in Haskell. In fact, HasCasl has an executable subset that corresponds quite closely to a large subset of Haskell. Features of HasCasl include higher-order types, type constructors, and parametric polymorphism. HasCasl is based on the partial $\lambda$-calculus so that various extensions to the logic can be formulated within the language itself, like general recursion. The semantics of a HasCasl-specification is given by a translation into a partial $\lambda$-theory. More precisely, the semantics of HasCasl is defined by a set-theoretic notion of intensional algebras. Thus, on the level of semantics HasCasl is very close to the approach of higher-order partial algebras presented in Section 3.1. The main difference is that in HasCasl $\lambda$-terms are taken into account. A detailed definition of HasCasl and its semantics is beyond the scope of this thesis and we refer to [Mos05]. In the following we focus on the language constructs and present several HasCasl-specifications, which can be used

for our concept of algebraic higher-order net and rule systems (see Section 6.2) and for our concept of higher-order rule systems (see Section 6.3). In detail we present HasCasl-specifications of P/T-systems, graphs, and rule-based transformations. The HasCasl-specifications involved have been checked with the Heterogeneous Tool Set [Het]. In this thesis we can only show a part of the involved specifications. All the specifications, especially for sets, partial maps, and multisets, can be found in [Lib].

In general, in HasCasl-specifications basic types are introduced by means of the keyword **type**. There are built-in type constructors $\_\_ \star \_\_$ for product types, $\_\_ - >? \_\_$ and $\_\_ - > \_\_$ for partial resp. total function types, **pred** for predicate types and a unit type *Unit*. A type can be formed by using the basic types and these type constructors. An operation is a constant of appropriate type. Higher-order terms are either variables, applications, tuples, or (multi-argument) $\lambda$-abstractions. Higher-order terms are used in axioms appearing in the form of conditional existence equation described in the previous section. Axioms may be universally quantified over type variables at the outermost level.

Fig. 7.1 and Fig. 7.2 are introducing a specification of P/T-nets and P/T-systems. These specifications more or less follow the definition given in Section 2.1. They rely on vocabularies for places and transitions. These vocabularies are given by types in the first place and later on (when forming the category) by sorts since a category needs to have definite sorts of objects and morphisms (type constructors are not involved). Actually the only freedom in the models is in the interpretation of these sorts. Typical choices will be the set of integers or strings. Once this choice has been made, the remaining parts of the models are determined uniquely up to isomorphism and hence a canonical model for the specification can be selected. The type of P/T-nets is then constructed by a set of places and two mappings for the pre- and post domain. The set of transitions is not explicitly specified. It is implicitly given by the domains of the pre- and post domain. These two mappings are rather partial than total function because of the given vocabulary for transitions. For this reason we need further axioms to ensure that the domains are identical. Otherwise, there may be a transition, where the pre domain is defined but the post domain is undefined and vice versa. Moreover, we have to make sure that the places in the codomain occur in the net. To achieve a specification of the category of P/T-nets, we give a specification of P/T-net morphisms. A P/T-net morphisms is combined by two P/T-nets and two (partial) functions for places resp. transitions. Here the ternary predicate $hp :: places\ n1 \longrightarrow places\ n2$ indicates for $hp : P - >? P$ and $p : Set\ P$ that $hp$, when restricted to $places\ n1$, actually yields results in $places\ n2$ and analogously for $ht :: transitions\ n1 \longrightarrow transitions\ n2$. Furthermore, P/T-net morphisms have to be compatible with pre- and post domains.

For the specification of P/T-systems we have to take markings into account. Markings are just multisets of places and a system is a net with an initial marking (such that the marking is actually only using the places of the net). The firing operation $\_\_[< \_\_>$ is only defined if the pre domain of the transition to be fired is contained in the current marking, and in this case, it just subtracts the pre domain and adds the post domain to the current marking.

Analogously we give a specification of graphs and graph homomorphisms (see Fig. 7.3).

We specify rule-based transformations via the double-pushout approach within the HasCasl-specification Transformation[Category] (see Fig. 7.4), where the parameter Category can be instantiate by arbitrary categories with a distinguished class of morphisms for which the construction of pushout complements can be uniquely obtained (examples can be found in [EHKP91, EP91, PER95]). Within the specification Category we have introduced an operation to obtain selected pushouts instead of pushouts up to isomorphisms for technical reasons. The

**spec** PETRINET = MAP **and** RICHMULTISET
**then**
**sorts** $P, T$
**type** $Net = \{(p, pre, post) :$
$Set\ P \times (T \to? MultiSet\ P) \times (T \to? MultiSet\ P)$
$\bullet\ dom\ pre = dom\ post$
$\wedge\ (\forall p1 : MultiSet\ P \bullet p1\ isIn\ range\ pre \Rightarrow MultiSetToSet\ p1 \subseteq p)$
$\wedge\ (\forall p1 : MultiSet\ P \bullet p1\ isIn\ range\ pre \Rightarrow MultiSetToSet\ p1 \subseteq p)\}$
**ops** $places : Net \to Set\ P;$
$transitions : Net \to Set\ T;$
$preMap, postMap : Net \to (T \to? MultiSet\ P)$
**forall** $n : Net;\ p : Set\ P;\ pre, post : T \to? MultiSet\ P$
$\bullet\ let\ n = (p, pre, post)\ in\ places\ n = p$
$\bullet\ let\ n = (p, pre, post)\ in\ transitions\ n = dom\ pre$
$\bullet\ let\ n = (p, pre, post)\ in\ preMap\ n = pre$
$\bullet\ let\ n = (p, pre, post)\ in\ postMap\ n = post$

**spec** PETRINETCATEGORY = PETRINET **and** MAPMULTISET
**then**
**sorts** $P, T$
**type** $HomNet = \{(n1, hp, ht, n2) :$
$Net \times (P \to? P) \times (T \to? T) \times Net$
$\bullet\ hp :: places\ n1 \longrightarrow places\ n2$
$\wedge\ ht :: transitions\ n1 \longrightarrow transitions\ n2$
$\wedge\ \forall t : T \bullet t\ isIn\ transitions\ n1 \Rightarrow$
$(freeMap\ hp(preMap\ n1\ t) = preMap\ n2(ht\ t)$
$\wedge\ freeMap\ hp(postMap\ n1\ t) = postMap\ n2(ht\ t))\}$
**ops** $dom : HomNet \to Net;$
$cod : HomNet \to Net;$
$placesMap : HomNet \to (P \to? P);$
$transitionsMap : HomNet \to (T \to? T);$
$id : Net \to? HomNet;$
$\_o\_ : HomNet \times HomNet \to? HomNet$
**pred** $injective : HomNet$
**forall** $n, n1, n2 : Net;\ hp : P \to? P;\ ht : T \to? T;\ h, h1, h2 : HomNet$
$\bullet\ let\ h = (n1, hp, ht, n2)\ in\ dom\ h = n1$
$\bullet\ let\ h = (n1, hp, ht, n2)\ in\ cod\ h = n2$
$\bullet\ let\ h = (n1, hp, ht, n2)\ in\ placesMap\ h = hp$
$\bullet\ let\ h = (n1, hp, ht, n2)\ in\ transitionsMap\ h = ht$
$\bullet\ id\ n = (n, \forall p : P \bullet p\ when\ p\ isIn\ places\ n\ else\ undefined,$
$\forall t : T \bullet t\ when\ t\ isIn\ transitions\ n\ else\ undefined, n)$
$as\ HomNet$
$\bullet\ def\ (h2\ o\ h1) \Leftrightarrow cod\ h1 = dom\ h2$
$\bullet\ def\ (h2\ o\ h1) \Rightarrow h2\ o\ h1 =$
$(dom\ h1, placesMap\ h2\ o\ placesMap\ h1,$
$transitionsMap\ h2\ o\ transitionsMap\ h1, cod\ h2)\ as\ HomNet$
$\bullet\ injective\ h \Leftrightarrow injective(placesMap\ h)$
$\wedge\ injective(transitionsMap\ h)$
**sort** $M = \{h : HomNet \bullet injective\ h\}$

Figure 7.1: Specification of P/T-nets in HASCASL

**spec** PETRISYSTEM = MAPMULTISET **and** PETRINET

**then**

**type** $Marking := MultiSet\ P$

**type** $System = \{(n, m) : Net \times Marking$
$\bullet\ let\ n = (p, pre1, post1)\ in$
$\forall x : P \bullet x\ isIn\ m \Rightarrow x\ isIn\ p\}$

**ops** $marking : System \to Marking;$
$net : System \to Net;$
$\_\_| < \_\_ > : System \times T \to System$

**forall** $sys, sys1, sys2 : System;\ n : Net;\ m : Marking;\ t : T;$
$\bullet\ let\ sys = (n, m)\ in\ net\ sys = n$
$\bullet\ let\ sys = (n, m)\ in\ marking\ sys = m$
$\bullet\ def\ sys| < t > \Leftrightarrow t\ isIn\ dom(preMap(net(sys)))$
$\wedge\ preMap(net(sys))\ t \leq marking(sys)$
$\bullet\ def\ sys| < t > \Rightarrow sys| < t > = (net(sys), (marking(sys))$
$-preMap(net(sys))\ t) +\ postMap(net(sys))\ t)$

**spec** PETRISYSTEMCATEGORY =
PETRISYSTEM **and** PETRINETCATEGORY **hide** $M$

**then**

**type** $HomSys = \{(sys1, hp, ht, sys2) :$
$System \times (P \to ?\ P) \times (T \to ?\ T) \times System$
$\bullet\ ((net(sys1), hp, ht, net(sys2))\ in\ HomNet)$
$\wedge\ \forall p : P \bullet freq(p, marking(sys1)) \leq freq(hp\ p, marking(sys2))\}$

**ops** $dom : HomSys \to System;$
$cod : HomSys \to System;$
$placesMap : HomSys \to (P \to ?\ P);$
$transitionsMap : HomSys \to (T \to ?\ T);$
$id : System \to ?\ HomSys;$
$\_o\_ : HomSys \times HomSys \to ?\ HomSys$

**pred** $injective : HomSys;$
$strict : HomSys$

**forall** $sys, sys1, sys2 : System;\ hp : P \to ?\ P;\ ht : T \to ?\ T;$
$h, h1, h2 : HomSys$
$\bullet\ let\ h = (sys1, hp, ht, sys2)\ in\ dom\ h = sys1$
$\bullet\ let\ h = (sys1, hp, ht, sys2)\ in\ cod\ h = sys2$
$\bullet\ let\ h = (sys1, hp, ht, sys2)\ in\ placesMap\ h = hp$
$\bullet\ let\ h = (sys1, hp, ht, sys2)\ in\ transitionsMap\ h = ht$
$\bullet\ id\ sys = (sys, \forall p : P \bullet p\ when\ p\ isIn\ places\ n\ else\ undefined,$
$\forall t : T \bullet t\ when\ t\ isIn\ transitions\ n\ else\ undefined,$
$sys)\ as\ HomSys$
$\bullet\ def\ (h2\ o\ h1) \Leftrightarrow cod\ h1 = dom\ h2$
$\bullet\ def\ (h2\ o\ h1) \Rightarrow h2\ o\ h1 =$
$(dom\ h1, placesMap\ h2\ o\ placesMap\ h1,$
$transitionsMap\ h2\ o\ transitionsMap\ h1, cod\ h2)\ as\ HomSys$
$\bullet\ injective\ h \Leftrightarrow injective(placesMap\ h)$
$\wedge\ injective(transitionsMap\ h)$
$\bullet\ let\ h = (sys1, hp, ht, sys2)\ in$
$strict\ h \Leftrightarrow injective\ h$
$\wedge\ \forall p : P \bullet freq(p, marking(sys1)) = freq(hp\ p, marking(sys2))$

**sort** $M = \{h : HomSys \bullet strict\ h\}$

Figure 7.2: Specification of P/T-systems in HASCASL

**spec** DIRECTEDGRAPH = MAP

**then**

**sorts** $N, E$

**type** $Graph = \{(n, source, target) :$
$Set\ N \times (E \to ?\ N) \times (E \to ?\ N)$
$\bullet\ dom\ source = dom\ target$
$(\forall n1 : Set\ N \bullet n1\ isIn\ range\ source \Rightarrow n1\ isIn\ n)$
$\wedge\ (\forall n1 : Set\ N \bullet n1\ isIn\ range\ target \Rightarrow n1\ isIn\ n)\}$

**ops** $nodes : Graph \to Set\ N;$
$edges : Graph \to Set\ E;$
$sourceMap, targetMap : Graph \to (E \to ?\ N)$

**forall** $g : Graph\ N\ E;\ n : Set\ N;\ source, target : E \to ?\ N$
$\bullet\ let\ g = (n, source, target)\ in\ nodes\ g = n$
$\bullet\ let\ g = (n, source, target)\ in\ edges\ g = dom\ source$
$\bullet\ let\ g = (n, source, target)\ in\ sourceMap\ g = source$
$\bullet\ let\ g = (n, source, target)\ in\ targetMap\ g = target$

**spec** GRAPHCATEGORY = DIRECTEDGRAPH **and** MAP

**then**

**sorts** $N, E$

**type** $HomGraph = \{(g1, hn, he, g2) :$
$Graph \times (N \to ?\ N) \times (E \to ?\ E) \times Graph$
$\bullet\ hn :: nodes\ g1 \longrightarrow nodes\ g2 \wedge he :: edges\ g1 \longrightarrow edges\ g2$
$\wedge\ \forall e : E\ \bullet\ e\ isIn\ edges\ g1 \Rightarrow$
$(hn(sourceMap\ g1\ e) = sourceMap\ g2(he\ e)$
$\wedge\ hn(targetMap\ g1\ e) = targetMap\ g2(he\ e))\}$

**ops** $dom : HomGraph \to Graph;$
$cod : HomGraph \to Graph;$
$nodeMap : HomGraph \to (N \to ?\ N);$
$edgeMap : HomGraph \to (E \to ?\ E);$
$id : Graph \to HomGraph;$
$\_o\_ : HomGraph \times HomGraph \to ?\ HomGraph$

**pred** $injective : HomGraph$

**forall** $g, g1, g2 : Graph;\ hn : N \to ?\ N;\ he : E \to ?\ E;$
$h, h1, h2 : HomGraph$
$\bullet\ let\ h = (g1, hn, he, g2)\ in\ dom\ h = g1$
$\bullet\ let\ h = (g1, hn, he, g2)\ in\ cod\ h = g2$
$\bullet\ let\ h = (g1, hn, he, g2)\ in\ nodeMap\ h = hn$
$\bullet\ let\ h = (g1, hn, he, g2)\ in\ edgeMap\ h = he$
$\bullet\ id\ g = (g, \forall n : N \bullet n\ when\ n\ isIn\ nodes\ g\ else\ undefined,$
$\forall e : E \bullet e\ when\ e\ isIn\ edges\ g\ else\ undefined, g)$
$as\ HomGraph$
$\bullet\ def\ (h2\ o\ h1) \Leftrightarrow cod\ h1 = dom\ h2$
$\bullet\ def\ (h2\ o\ h1) \Rightarrow h2\ o\ h1 =$
$(dom\ h1, nodeMap\ h2\ o\ nodeMap\ h1,$
$edgeMap\ h2\ o\ edgeMap\ h1, cod\ h2)\ as\ HomGraph$
$\bullet\ injective\ h \Leftrightarrow injective(nodeMap\ h) \wedge injective(edgeMap\ h)$

**sort** $M = \{h : HomGraph \bullet injective\ h\}$

Figure 7.3: Specification of graphs in HASCASL

notation $M < Mor$ in Fig. 7.4 introduces $M$ as a subsort (roughly corresponding to a subset) of sort *Mor*. Based on the predicate *POComplement*, which states that there is a construction of pushout complements, we specify the operation *transform* for rule-based transformations. Note that *transform* is a partial function and *def transform(p,g1)* states that a rule *p* is applicable with occurrence morphism *g1*. The first axiom in Fig. 7.4 specifies the domain of definition for *transform*, while the second axiom specifies its effect when defined. Due to the first axiom the pushout construction has to be uniquely determined. For our purpose, it is sufficient to note that for the following examples of categories it has been checked that there is a unique construction of pushout complements.

- $(\mathbf{PTNets}, \mathcal{M}_{injective})$ where **PTNets** is the category of P/T-nets and the class $\mathcal{M}_{injective}$ of injective Petri net morphisms (see [EHKP91]),
- $(\mathbf{PTSys}, \mathcal{M}_{strict})$ where **PTSys** is the category of P/T- systems, i.e. P/T-nets together with an initial marking where the class $\mathcal{M}_{strict}$ consists of strict inclusions so that the marking is preserved (see Section 2.1), and
- $(\mathbf{Graph}, \mathcal{M}_{injective})$ where **Graph** is the category of coloured graphs and the class $\mathcal{M}_{injective}$ of injective, colour preserving graph morphisms (see [EHKP91]).

Thus, we can give the following instantiations of the HasCasl-specification Category and obtain the notions of rules and an operation for rule-based transformations for each category mentioned above, especially for P/T-systems according to the definitions in Section 2.1.

- Transformation[PetriNetCategory]
  for P/T-nets and P/T-net rules using the specification in Fig. 7.1
- Transformation[PetriSystemCategory]
  for P/T-systems and P/T-system rules using the specification in Fig. 7.2
- Transformation[GraphCategory]
  for directed graphs and graph rules using the specification in Fig. 7.3

Based on the specification of Transformation[HLRCategory] we specify the concept of inheritance to define new rules by reusing existing rules (see Section 6.3) within the HasCasl-specification LocalTransformation[HLRCategory] (see Fig. 7.2).

In [HM02, HMPP04] we have presented the formalism of AHO-nets, where in contrast to the definitions of AHO-nets in Chapter 3 the specifications in HasCasl as described above are used. As mentioned in the introduction of this chapter there are several reasons to use higher-order signatures here instead of higher-order specifications for our concept. But the combination of HasCasl-specifications and Petri nets is a promising topic of further research because HasCasl is a formal approach for functional programming languages and tools for HasCasl already have been implemented.

**spec** HLRCategory = Pushout[Category]

**then**

**sort** $M < Mor$
**reveal** ***sorts*** $Ob, Mor, M$, ***ops*** $id, dom, cod, \_o\_$

**spec** Transformation[HLRCategory] = Pushout[Category]
**then**

**type** $Rules = \{(l, r) : M \times M \bullet dom\ l = dom\ r\}$

**ops** $transform : Rules \times Mor \to ? Ob;$

**pred** $POComplement : Mor \times Mor \times Ob$

**forall** $o : Ob;\ f, h, g1 : Mor;\ p : Rules$

- $POComplement(f, h, o) \Leftrightarrow$
  $\exists g, k : Mor \bullet (h, k) = f\ pushout\ g\ \wedge\ dom\ k = o$
- $let\ (l, r) = p\ \ in$
  $def\ transform(p, g1) \Leftrightarrow$
  $\exists g2, g3, c1, c2 : Mor \bullet POComplement(l, g1, dom\ c1)$
  $\wedge\ (\forall o1, o2 : Ob \bullet$
  $POComplement(l, g1, o1)$
  $\wedge POComplement(l, g1, o2) \Rightarrow o1 = o2)$
  $\wedge\ (g1, c1) = l\ pushout\ g2$
  $\wedge\ (g3, c2) = r\ pushout\ g2$
- $let\ (l, r) = p\ \ in$
  $def\ transform(p, g1) \Rightarrow$
  $\exists g2, g3, c1, c2 : Mor \bullet$
  $\wedge\ (g1, c1) = l\ pushout\ g2 \wedge\ (g3, c2) = r\ pushout\ g2$
  $\wedge\ transform(p, g1) = cod\ g3$

**view** CategoryofPetriNets : HLRCategory **to** PetriNetCategory =
$Ob \mapsto Net, Mor \mapsto HomNet, \_o\_, dom, cod, id, M$

**view** CategoryofPetriSystems : HLRCategory **to**
PetriSystemCategory =
$Ob \mapsto System, Mor \mapsto HomSys, \_o\_, dom, cod, id, M$

**view** CategoryofGraphs : HLRCategory **to** GraphCategory =
$Ob \mapsto Graph, Mor \mapsto HomGraph, \_o\_, dom, cod, id, M$

Figure 7.4: Specification of rule-based transformations in HasCasl

**spec** LocalTransformation[HLRCategory] =

Transformation[HLRCategory]

**then**

**ops** $inherit : Rules \times Rules \times Mor \to ? Rules$;

**forall** $p1, p : Rules;\ g1 : Mor$

- $let\ (l1, r1) = p1\ \wedge\ (l, r) = p\ \text{in}$
  $def\ inherit(p1, p, g1) \Leftrightarrow cod\ g1 = cod\ r1$
  $\wedge\ def\ transform(p, g1)$
  $\wedge\ def\ r1\ pullback\ c1$
- $let\ (l1, r1) = p1\ \wedge\ (l, r) = p\ \text{in}$
  $def\ inherit(p1, p, g1) \Rightarrow$
  $\exists g4, g5, c1, c2 : Mor \bullet transform(p, g1) = (c1, c2)$
  $\wedge\ r1\ pullback\ c1 = (g4, g5)$
  $\wedge\ inherit(p1, p, g1) = (l1\ o\ g4, c2\ o\ g5)$

Figure 7.5: Specification of inheritance in HasCasl

# Chapter 8

# Case Study Medical Information System

In this chapter we demonstrate the practical relevance of our new concept of algebraic higher-order net and rule systems introduced in Section 6.2 by a large case study in the area of medical information systems. This case study is inspired by the case study proposal on hospital therapeutic processes in [Han97]. Although this proposal is leaving some room for interpretation, it pinpoints a baseline process concerning the receiving and curing of patients. In this thesis we present the main part of the case study [Kie04]. A first sketch has been published in [HM02].

## 8.1 Introduction

The case study "Medical Information System" deals with some business processes in a hospital, in particular patient therapeutic treatments. The hospital consists of two specialized departments. In detail, there is one department for orthopedic diseases and one intensive care unit. The hospital also has a reception office, which organizes the admission and the discharge to and from the hospital, and a laboratory department, which is responsible for various tests and specific technical treatments. Each department is rather independently organized, i.e. they have their own staff, equipments, and internal activities. The idea is to model the following situation. First, patients are received at the reception office. In the normal case, the patient has been sent to the hospital by a doctor's order, while in the emergency case the patient has to be immediately treated before the formal admission in the hospital could start. However, during the registration patient documents are created by taking the patient record and starting the initial patient care plans. After a diagnosis is made, it will be decided which department is responsible for the patient and whether the patient should receive in-patient or out-patient treatments. Afterwards, specialists prescribe medication and certain specific and general treatments like x-ray examinations or measuring blood pressure, which are recorded in the corresponding patient care plan. Care plans have to be constantly modified according to the treatment effects, for instance effectiveness of medications, and have to be carried out, if demanded.

Summarizing, the idea of the case study "Medical Information System" is to model the coordination of patient care plans within the hospital, in detail the initialization, extension, transformation, and execution of patient care plans. Hence, our model is realized using the concept of algebraic higher-order net and rule systems. The organizational structure of the hospital is reflected in the system level given by an AHO-net, while patient care plans are modeled by dynamic tokens given

by P/T-systems. Because patient care plans are not fixed once and for all, we use rules as a specific kind of resources for the transformation of patient care plans.

## 8.2 Data Type Part

Let us point out that the data type part of the case study "Medical Information System" given in the master thesis [Kie04] is defined by the HasCasl-specification Hospital, which is an extension of the HasCasl-specifications of P/T-systems and rules as presented in Section 7.2 especially for this case study. In this thesis we sketch the main part of the HasCasl-specification Hospital and informally explain the intuition of the data type part in the subsequent sections.

The HasCasl-specification Hospital provides additional types *waiting, ambulant* and *intensiv* for specific therapeutic treatments, which have to be carried out in the laboratory department, the in-patient resp. out-patient departments. The patient record called *ID* consists of the first and last name together with the date of birth. Moreover, there are some constants, $reception_1 \ldots$, $reception_4$, for initial patient care plans. The P/T-systems, which correspond to these constants, are depicted in Fig. 8.2 - Fig. 8.5. To support the flexibility of the model, we introduce a further operation *ident*, which takes a rule and a morphism as parameters but leaves the patient care plans unchanged. As we will see, this operation is useful to achieve a more compact model. Patient care plans are rather workflow systems than arbitrary P/T-systems, i.e. they are equipped with a "start" place and a "stop" place and are in some sense strongly connected. For this reason we have to ensure that patient care plans can be carried out, which is called liveness in terms of Petri nets. Liveness of systems means that no deadlock and even livelock of a system can occur, i.e. there always exists a firing sequence which enables any chosen transition from any reachable marking. Thus, there is a specific predicate *isLive* to check whether a P/T-system is still live after the application of a rule. For the same reason there is an operation *switchableTrans* to compute the set of transitions, which are enabled in a P/T-system. Finally, the operation *objCoproduct* computes the disjoint union of two P/T-systems.

## 8.3 System Level

In Fig. 8.6 we present the system level of our case study "Medical Information System". The system level is given by an algebraic higher-order net and rule (AHONR) system explained in Section 6.2. Here we omit the detailed net inscriptions to give a rough idea about the whole model. The system level consists of three different parts. There is on the one hand the reception office and on the other hand the department A for orthopedic diseases and the department B, which is the intensive care unit. The department A is divided into two parts for in-patient and out-patient treatments.

A marking of the AHONR-system represents the distribution of patients at different places in the hospital and the firing behavior of the AHONR-system describes the admission, the curing and discharge of patients. There are different locations in the hospital where patients can stay: reception, ward A and ward B. Each location is represented by several places in the AHONR-system in Fig. 8.6. Initially several patients are waiting at the reception office on place *patient*.

Patients may move around, which means they might leave and enter the reception office and they might leave and enter the department A resp. the department B. The mobility aspect of patients is modeled by several transitions in the AHONR-system in Fig. 8.6 like *admit in department A*. While patients are moving around,

**spec** HOSPITAL = PETRISYSTEMCATEGORY

**and** TRANSFORMATION[PETRINETCATEGORY]
**and** LIST[CHAR]

**then**

**type** $waiting := Set\ T$
$ambulant := Set\ T$
$intensiv := Set\ T$
**type** $string := List[Char]$
**type** $ID = \{(name, firstname, birthdate) : string \times string \times nat$
**ops** $reception1 : System;$
$reception2 : System;$
$reception3 : System;$
$reception4 : System;$
$indent : Rules \times Mor \to System;$
$switchableTrans : System \to Set\ T;$
$objCoproduct : System \times System \to System;$
**pred** $isLive : System;$

$\vdots$

Figure 8.1: Specification Hospital in HASCASL

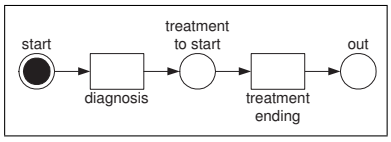



Figure 8.2: Initial singular care plan *reception 1*

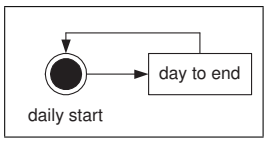



Figure 8.3: Initial daily care plan *reception 2*

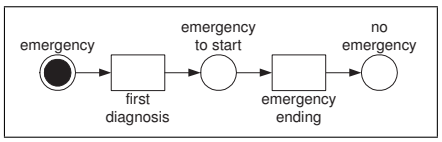



Figure 8.4: Initial emergency care plan *reception 3*

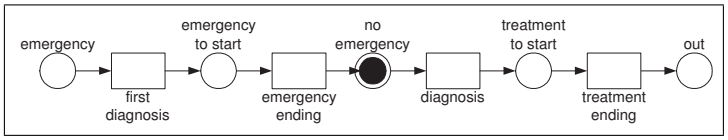



Figure 8.5: Initial singular care plan *reception 4*

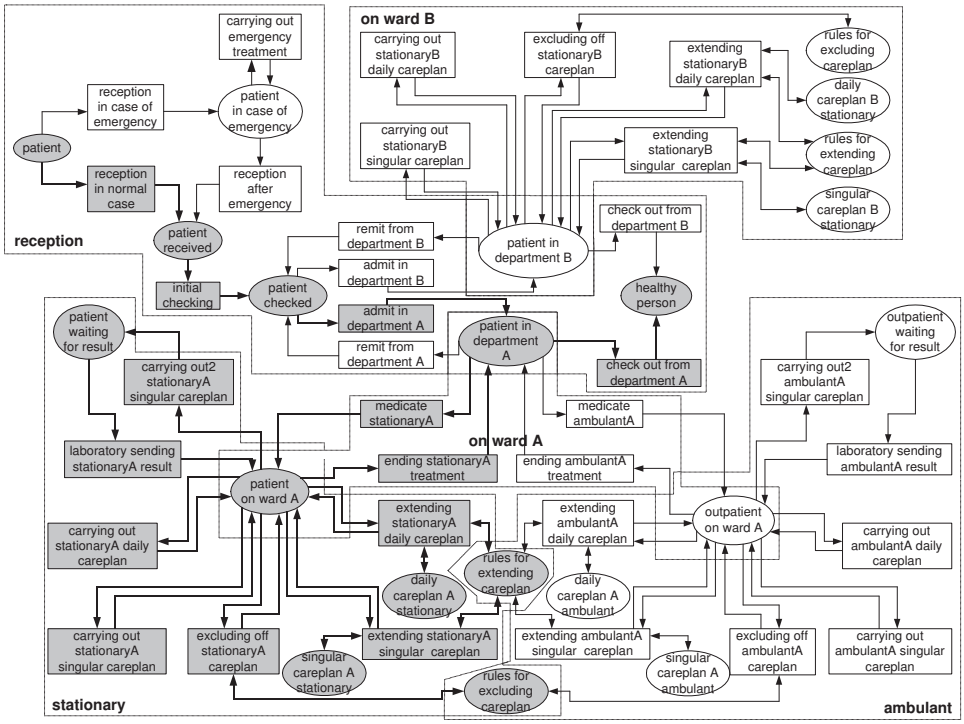



Figure 8.6: System level (without net inscriptions)

the static structure of their patient care plans are changed by rule-based transformations. For this reason we have rule tokens, which are used as resources and deposited on several places in the AHONR-system in Fig. 8.6 like *rules for extending care plans*. Because patient care plans are modeled by P/T-systems, they have their own internal behavior. To realize different markings of P/T-systems there are several transitions in our system like *carrying out stationaryA singular care plan*.

In the following we focus on the reception office and the in-patient treatments in the department A. Department B is in some sense similar, i.e. it has more or less the same net structure as department A but different treatments can be carried out.

In Fig. 8.7 the system level of the reception office is depicted in more detail. The reception office organizes the admission to the hospital and the discharge from the hospital. In the normal case patient documents are created by using the patient ID consisting of the first and last name and the date of birth. Moreover initial patient care plans are started. The initial patient care plan consists of the care plan *reception 2* for daily treatments (see Fig. 8.3) and the care plan *reception 1* for singular treatments (see Fig. 8.2).

In the emergency case patients have to be immediately treated before the formal admission, i.e. the initial care plan consists of the patient care plan *reception 3* in Fig. 8.4. To start the emergency treatments we use the transition *carrying out emergency treatment* of the AHONR-system in Fig. 8.7. We have to give a variable valuation, where the variable $n_1$ is assigned to the P/T-system *reception 2* and the variable $t$ to the transition *first diagnosis*. The evaluation of the term $n_1| < t >$ in the post domain of the transition *carrying out emergency treatment* computes the follower marking of the P/T-system *reception 3*, i.e. token *emergency to start*. Now the transition *emergency ending* in the P/T-system *reception 3* in Fig. 8.4 is activated. In a second step we compute the follower marking, i.e. token *no emer-*

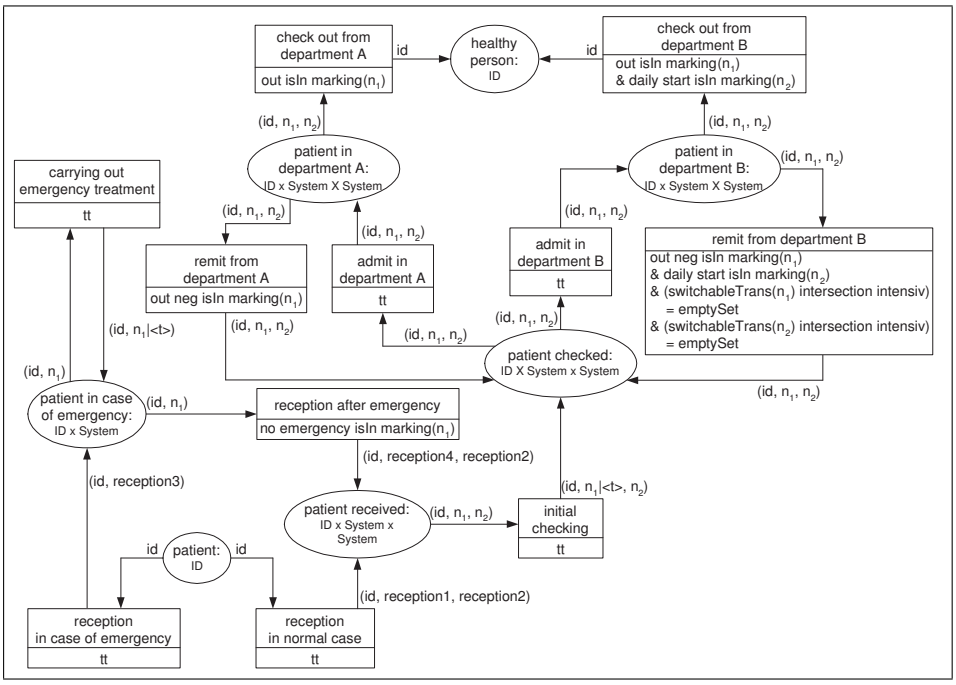



Figure 8.7: Reception office

*gency*, in the same way as described above but using a different variable valuation, where we assign the transition *emergency ending* to the variable $t$. Afterwards the patient has been received in the hospital. Here we use the transition *reception after emergency* of the AHONR-system in Fig. 8.7 to start the initial patient care plan consisting of the singular care plan *reception 4* in Fig. 8.5 and the daily care plan *reception 2* in Fig. 8.3.

It has to be decided which department is responsible for the patient. For this reason a diagnosis is made by using the transition *initial check* of the AHONR-system in Fig. 8.7. In any case the transition *diagnosis* in the singular care plan of a patient is enabled. We compute the follower marking as described above but assign the transition *diagnosis* to the variable $t$. Afterwards the initial singular care plan is in the state *treatments to start* (see Fig. 8.2 and Fig. 8.5). To admit patients to the departments A or B we use one of the transitions *admit in department A* and *admit in department B* of the AHONR-system in Fig. 8.7.

Note that under certain conditions patients can be remitted from these department. In the case of department A, the firing conditions of the transition *remit from department A* of the AHONR-system in Fig. 8.7 ensures, that the curing process does not stopped. In the case of department B the remission of a patient is more complicated because department B is the intensive care unit and specific treatments can only be carried out in this department. The firing conditions of the transition *remit from department B* of the AHONR-system in Fig. 8.7 ensures that daily treatments are carried out and transitions, which are enabled in corresponding care plans, are not specific for the intensive care unit. The last condition is notated by the net inscription $switchableTrans(n_1)$ *intersection intensiv* = *emptySet*, where the evaluation of the term $switchableTrans(n_1)$ computes the set of transitions, which are enabled in singular patient care plans, and the evaluation of the term *intensiv* corresponds to the set of specific treatments for the intensive care unit.

If the curing process gets stopped but the daily treatments have to be still carried out, the remission of patients is obtained by the transitions *check out from*

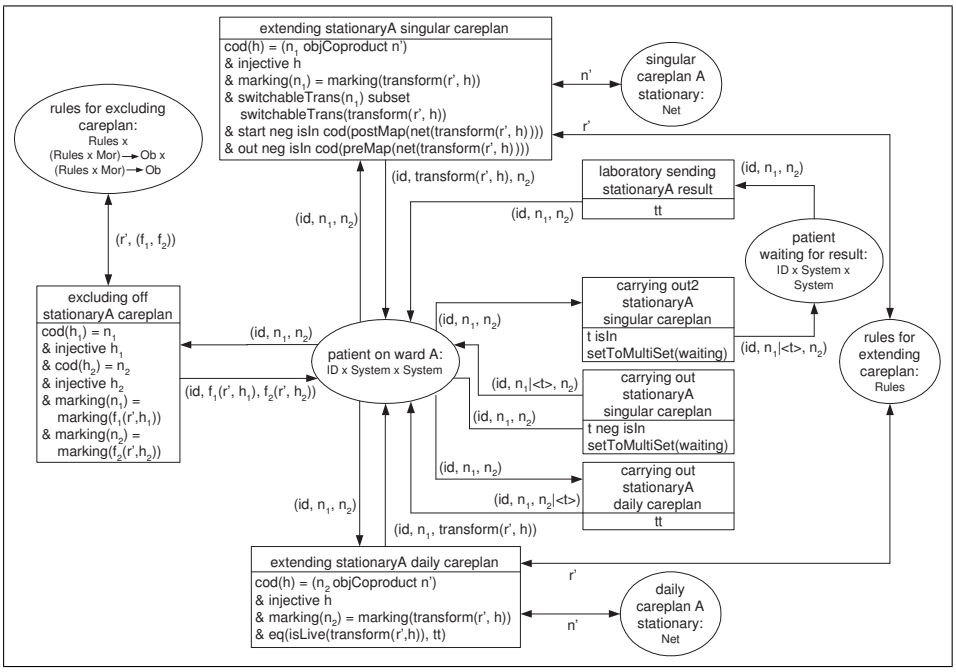



Figure 8.8: In-patient treatments of department A

Herrmann, Klaus,13071952

Figure 8.9: Patient ID

*department A* resp. *check out from department B* of the AHONR-system in Fig. 8.7.

In Fig. 8.8 the system level of the in-patient treatments of department A is depicted. The AHONR-system of department A consists of two different kinds of transitions. There are some transitions to carry out patient care plans, while other transitions reflect the transformation of patient care plans. Before we are able to explain the AHONR-system of department A in more detail, we have to speak about rules and treatments specific for our system.

## 8.4 Token Level

The token level consists of four different types of tokens: patient ID's, P/T-systems representing patient care plans, P/T-nets representing specific care plans and rules for the transformation of patient care plans.

Patient ID's are used to assign care plans to particular patients. An example is given in Fig. 8.9. For simplicity reasons a patient ID consists of the first and last name and the date of birth. But we can think about more complex identification features, like patient's addresses and information about medical insurance companies responsible for the invoice.

For each department we distinguish several care plans, which are specific for them. Each care plan is given by its own P/T-system with an empty initial marking. In our case study these P/T-systems consist of two places and one transition. The places indicate that a patient is before resp. after a treatment, while the transition is labeled by a specific treatment. These transitions can be seen as black boxes for a

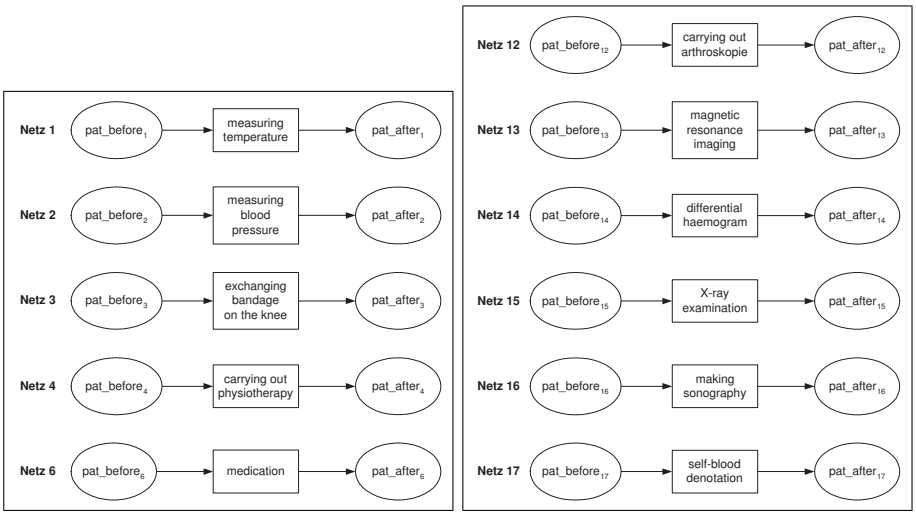



Figure 8.10: Daily and singular care plans of department A

number of separate treatments, which might be carried out in sequential and/or in parallel. The daily and singular care plans for in-patient treatments of department A are depicted in Fig. 8.10. These care plans are considered as tokens being in the place *singular careplan A stationary* resp. the place *daily careplan A stationary* of the AHONR-system in Fig. 8.8. Out-patient treatments of department A and specific treatments of department B can be found in [Kie04].

For the transformation of patient care plans we introduce different kinds of rules. These rules do not depend on specific care plans. They are general in the sense that they are classified according to their effects. On the one hand there are rules for the extension of patient care plans and on the other hand there are rules, which remove some treatments. Moreover, patient care plans can be extended by sequential and parallel treatments.

The sequential extension of patient care plans is realized by the rule shown on the left hand side of Fig. 8.11. The application of this rule introduces the treatment $t$ before the treatment $t1$. Because there can be more than one place $p1$ in the pre domain of the treatment $t1$ (and analogously in the post domain), we use the concept of rule schemes, where the number of places in the pre- resp. post domain is denoted by parameters. The rule scheme for the sequential extension is shown on the right hand side of Fig. 8.11, where the parameters are given by the variables $n$ resp. $m$. For the sequential extension of the initial daily care plan *reception 2* we need a particular rule (see Fig. 8.12), because we consider injective matches to avoid ambiguities. For instance we have to preserve the sequential net structure in singular care plans. The rules and rule scheme for the parallel extension are depicted in Fig. 8.13 and Fig. 8.14. These rules are represented as tokens in the place *rules for extending care plan* of the AHONR-system in Fig. 8.8.

Analogously, sequential and parallel treatment can be removed from patient care plans (see Fig. 8.15 - Fig. 8.20). These rules are represented as tokens in the place *rules for excluding care plan* of the AHONR-system in Fig. 8.8.

## 8.5 Process

To illustrate the semantics of our system in more detail, one of many possible algebraic higher-order occurrence nets (see Section 4.3) is depicted in Fig. 8.21. For simplicity reasons we omit firing conditions and typing of places. For a more

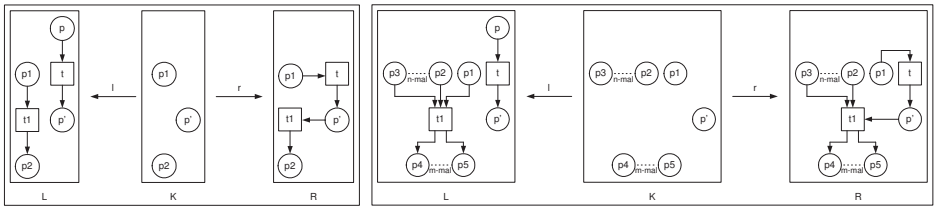



Figure 8.11: Rule and rule-scheme for sequential extensions

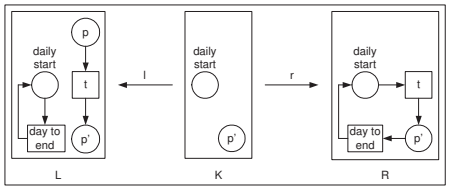



Figure 8.12: Rule for sequential extension in initial daily care plan

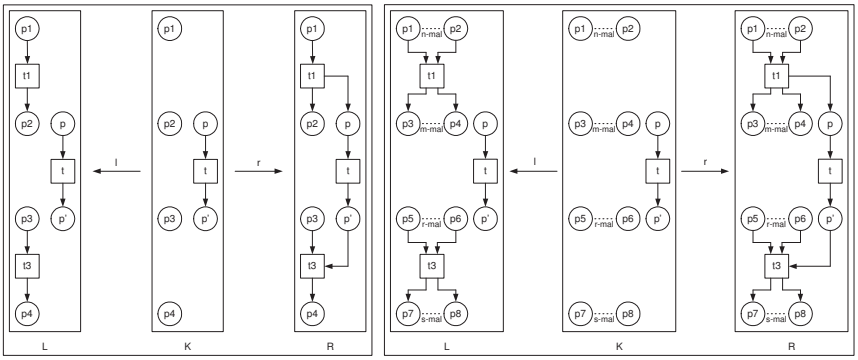



Figure 8.13: Rule and rule-scheme for parallel extension of care plans

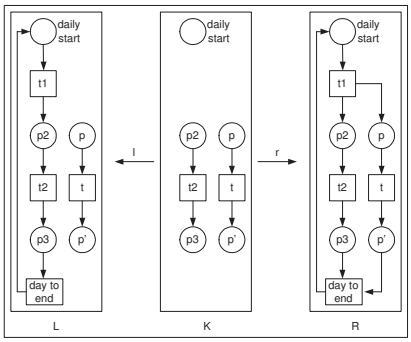



Figure 8.14: Rule for parallel extension in initial daily care plan

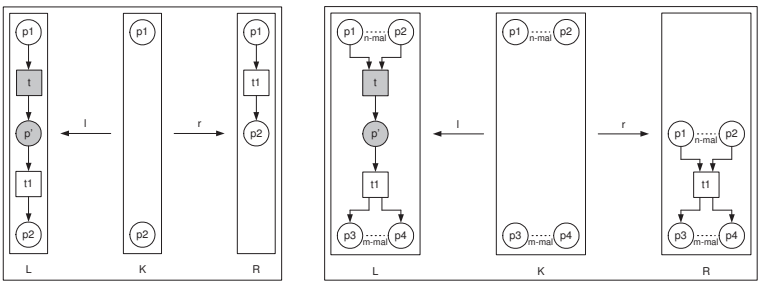



Figure 8.15: Rule and rule-scheme for sequential removal

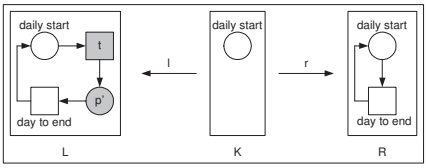



Figure 8.16: Rule for sequential removal in initial daily care plan

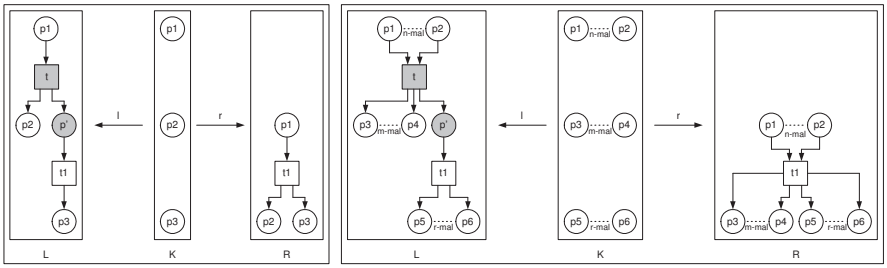



Figure 8.17: Rule and rule-scheme I for parallel removal

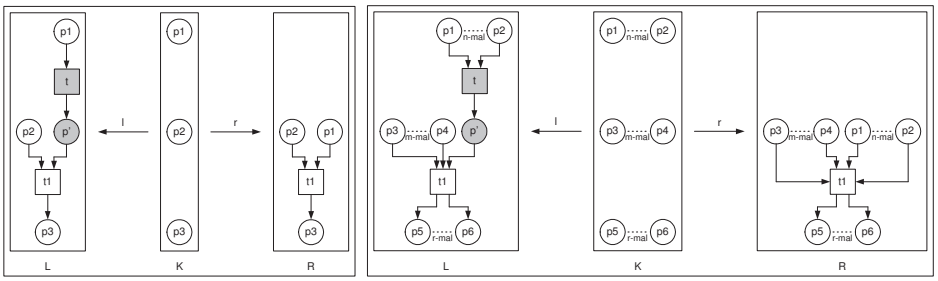



Figure 8.18: Rule and rule-scheme II for parallel removal

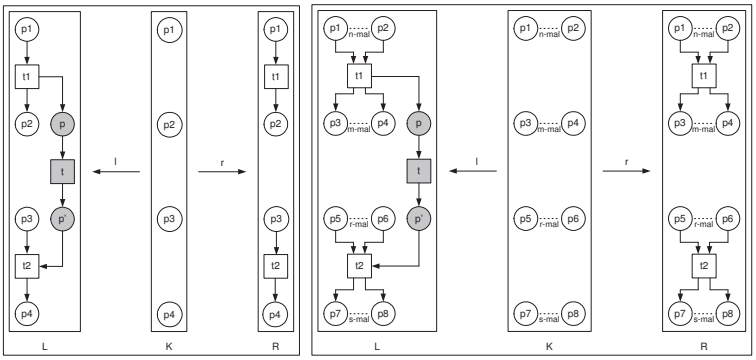



Figure 8.19: Rule and rule-scheme III for parallel removal

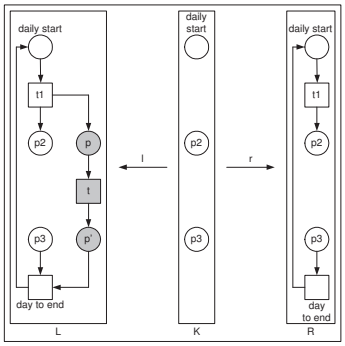



Figure 8.20: Rule for parallel removal in initial daily care plan

complex occurrence net we refer to [Kie04]. To obtain an algebraic higher-order process we have to define an algebraic higher-order net morphism, which maps the algebraic higher-order occurrence net in Fig. 8.21 into the AHONR-system in Fig. 8.8. For instance the places *patient on ward A*, ..., *patient on ward A""* in Fig. 8.21 are mapped to the place *patient on ward* in Fig. 8.8.

In the following we explain one firing sequence of the algebraic higher-order occurrence net in more detail. We assume that the token in Fig. 8.22 is on the place *patient on ward A* in Fig. 8.21. Moreover, let the rule in Fig. 8.11 for sequential extensions be on the place *rules for extending careplan* and the care plan *Netz 2* in Fig. 8.10 for measuring the blood pressure on the place *daily careplanA stationary*. We have to give an assignment $v$ for the variables of the transition *extending stationaryA daily care plan*, i.e. we assign the token in Fig. 8.22 to $(id, n_1, n_2)$, the rule in Fig. 8.11 to the variable $r'$, and the care plan *Netz 2* in Fig. 8.10 to the variable $n'$. Then we compute the disjoint union of the daily care plan at the bottom of Fig. 8.22 and the care plan *Netz 2* denoted by the net inscription $n_1$ *objCoproduct* $n'$ in the firing condition of the transition *extending stationaryA daily care plan* in Fig. 8.8. The result is the disjoint union of both systems.

In a next step we assign a suitable match morphism $g : L \to G$ to the variable $h$ so that the domain of $g$ is the left hand side of the rule in Fig. 8.11 and the codomain of $g$ is the disjoint union computed above. In detail we map the transition $t$ in the left hand side of the rule in Fig. 8.11 to the transition *measuring blood pressure* of *Netz 2* and the transition $t1$ to the transition *day to end* of the daily patient care plan in Fig. 8.22. We are able to apply the rule at the match $g$ denoted by the net inscription $transform(r, h)$, which results in the daily patient care plan depicted at the bottom in Fig. 8.23. The firing conditions of the transition *extending stationaryA daily care plan* in Fig. 8.8 are satisfied because the match $g$ is injective, the marking of daily care plan is still the same after the transformation, and the daily care plan after the transformation is obviously live.

Afterwards we extend the daily care plan at the bottom in Fig. 8.23 in the same way as above, but we use the rules for the parallel extension (see Fig. 8.13) and the care plan *Netz 1* for measuring the temperature (see Fig. 8.10). This results in the daily care plan at the bottom of Fig. 8.24.

The daily patient care plan at the bottom in Fig. 8.24 is carried out by firing the transition *carrying out stationaryA daily careplan* in Fig. 8.21. First the net inscription $(id, n_1, n_2)$ is assigned to the token in Fig. 8.24 and the variable $t$ to the transition *giving heparin injection* of the daily care plan. The evaluation of the term $n_2| < t >$ computes the follower marking of the daily care plan, i.e. token $pat\_after_5$ (see Fig. 8.25).

Finally, we want to remove the treatment *measuring blood pressure* from the daily care plan. Here we use the rule in Fig. 8.15. We assign the token in Fig. 8.25 to $(id, n_1, n_2)$ and the rule in Fig. 8.15 to the variable $r$. But we also have to give suitable operations to the variables $f_1$ and $f_2$. Because we want to transform the daily care plan but not the singular care plan, we assign the operation $transform$ to the variable $f_2$ and the operation *ident* to the variable $f_1$, which exactly realizes the intended behavior. Moreover, we have to give a suitable match morphism $h_2$, i.e. the transition $t$ in the left hand side of the rule in Fig. 8.15 is mapped to the transition *measuring blood pressure* of the daily care plan. The firing of the transition *excluding off stationaryA careplan* in Fig. 8.21 results in the patient care plan depicted in Fig. 8.26.

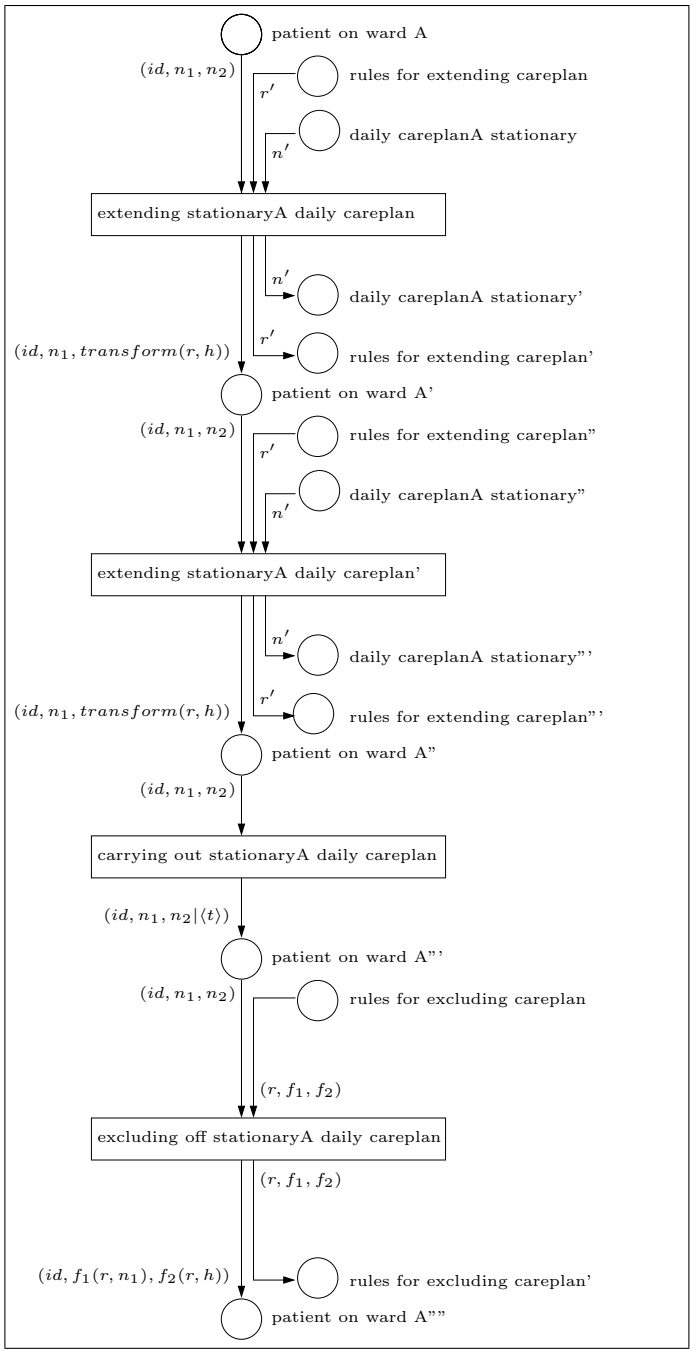



Figure 8.21: Algebraic higher-order occurrence net

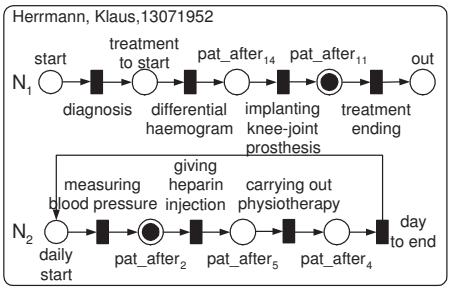



Figure 8.22: Marking of *patient on ward A*

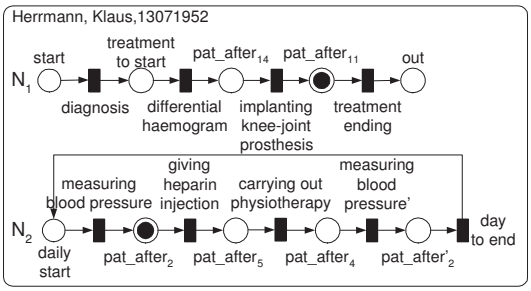



Figure 8.23: Marking of *patient on ward A'*

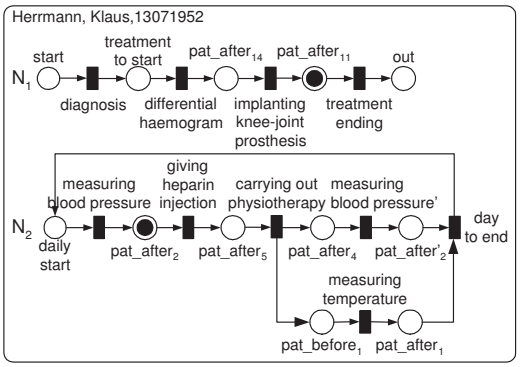



Figure 8.24: Marking of *patient on ward A"*

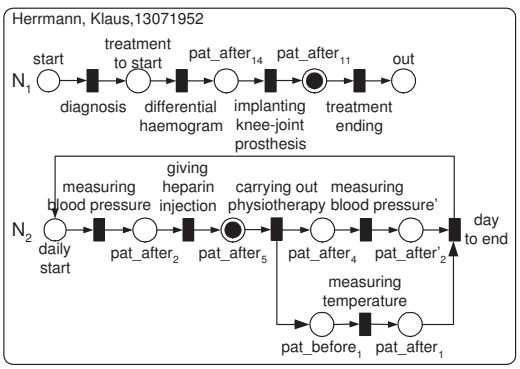



Figure 8.25: Marking of *patient on ward A"'*

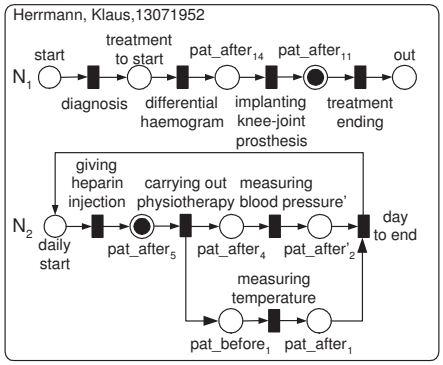



Figure 8.26: Marking of *patient on ward A""'*

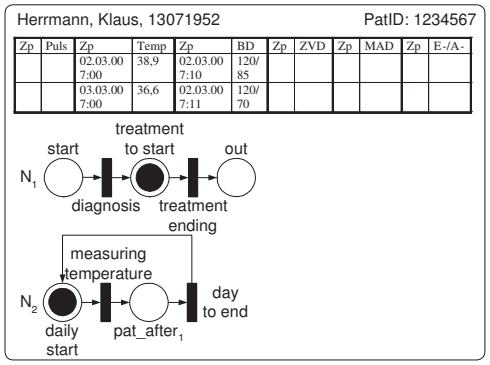



Figure 8.27: Combination of patient record and patient care plan

## 8.6 Further Aspects

In our case study "Medical Information System" there is a set of rules as resources for the transformation of patient care plans. Thus, transformations become effectively included into the system. The advantage of our approach is a more flexible modeling technique because we can add further token rules to realize other kinds of transformations, while the AHONR-system in Fig. 8.6 is fixed.

In our case study a patient care plan consists of a singular patient care plan and a daily patient care plan. An interest aspect is the extension of patient care plans by a more complex patient record, which includes for instance a prescription sheet and a temperature chart. Here we can use the results obtained by the case study "Distributed Information Management System (HDMS)" [Erm96]. This case study concentrates on the concept and development of electronic patient records, which arise during the stay of a patient at the hospital. The patient records are formalized by classical algebraic specifications (see Section 2.2). Thus, we can adapt these specifications to combine patient records and our approach of patient care plans. An example of such a combination is given in Fig. 8.27.

# Chapter 9

# Case Study Logistics

In this chapter we demonstrate the notions and results of this thesis in form of a case study in the area of logistics processes. The case study "Logistics" is based on data structures defined in [Sch94] by entity/relationship-diagrams and processes modeled by event driven process chains. The business process logistics consists of the planning and scheduling functions concerned with the distribution of products to customers. A reduced version of the case study "Logistics" has been published in [Hof03]. Here we present the main part of [Bog04] concerning algebraic higher-order nets. For the entire description we refer to the master thesis [Bog04], where the development process of the logistics process is formally investigated by net class transformations to change the underlying modeling formalism and net model transformations to refine a specific model. Thus, on the one hand models are enhanced by additional aspects like a suitable data structure, and on the other hand descriptions of models are refined by adding some details, like further exceptions.

First we present a low-level version of our case study to discuss the aims of the logistics process, especially the included activities. Next we enrich our model by a suitable data type part, which is not given explicitly in this thesis but can be found in [Bog04]. We just mention some important types and operations. Subsequently we demonstrate that we achieve not only a more compact description of the model but also a more flexible and abstract model using the concepts of horizontal structuring techniques (see Chapter 4) and folding constructions (see Chapter 5) within the case study "Logistics". In more detail we apply several times the folding construction wrt. constant symbols, the folding construction wrt. product types and the horizontal structuring technique fusion. Finally, we use the horizontal structuring technique union to obtain the overall system. As a conclusion we discuss interesting aspects of future work concerning rule-based transformations and their compatibility with net class transformations and horizontal structuring techniques.

## 9.1 Introduction

In view of departments the business process can be divided into the following five different parts, each of them with specific documents and activities.

- "Offer Preparation",
- "Order Acceptance",
- "Order Processing",
- "Shipping", and
- "Accounts Receivable".

In Fig. 9.1 the logistics process is modeled by a P/T-net, where the dashed lines indicate the subprocesses carried out in each department. In the following we describe the activities in each department in more detail. The tender preparation starts with a request of the customer. Not only customers, which are already known to the company, but also new customers should have the ability to send a request. In both cases an offer is prepared and send to the customer. If the customer accepts the offer, an order is generated using the information from the offer. Then the availability of ordered articles is checked by comparing the list of articles, which are available at the moment, and the list of ordered articles. The order may be split into two parts: the current order including all articles, which are available at the moment, and a new order including all remaining articles to be carried out as soon as they are available. The articles available are removed from the stock and provided for loading up on trucks. Furthermore the corresponding delivery note is generated to inform the customer about articles to be received and the articles are send to the customer. In a next step the invoice is generated and compared with the receipted delivery note. Finally, invoices are captured and cleared with incoming payments. Once a week outstanding payments are checked and the account department reminds the customers concerned.

## 9.2 Logistics Process as AHO-Net Scheme

To achieve a higher-order version of our case study "Logistics" there is a data type part for documents and activities included in the logistics process. It results in a very complex description of the data structure "Customer", which is depicted in Fig. 9.2 by the uses-hierarchy of the involved data structures. Besides the basic data types for articles, transport resources, actual dates and customer records, the data structures specific for each department are built up to reflect the information flow of the logistics process. For instance the data structure "Customer Offer" uses the information given in "Customer Request".

Our case study "Logistics" consists of five algebraic higher-order net schemes corresponding to the departments described above. They are depicted in the left upper corners of Fig. 9.3 - Fig. 9.7. In the following we describe the AHO-net scheme for the preparation of an offer in Fig. 9.3 in more detail. The AHO-net scheme consists of six places for customer records and requests of customers as well as for their offers. Moreover, there are three transitions to receive a request of a customer and to prepare an offer. A request consists of the name and the address of the customer as well as of a list of articles. The request of a new customer is enriched by an identification number and the customer record is stored in the customer data base. If the customer is known to the company, the customer record is compared with its entry in the customer data base. Afterwards a request is generated using the operation *mk_request*, which combines the identification number, the actual date, the list of requested articles and additional information. Then an offer is generated by the operation *mk_offer*, which mainly adds the identification number and the actual prices of the requested articles.

Because the involved operations are formalized by constants of appropriate higher-order type, we apply the folding construction wrt. constant symbols given in Section 5.1. For instance, there is the constant symbol *mk_offer* of function type $customer\ request \times date \rightarrow customer\ offer$ in the environment of the transition *make offer*. We assume that there is a suitable variable $f$ available in our data type part, which corresponds to this constant symbol. Due to the application of the folding construction we replace the constant symbol *make offer* by the variable $f$ in the post domain of the transition *make offer*. Moreover, we add a new place *P2*, where the assigned function type is given by $customer\ request \times date \rightarrow customer$

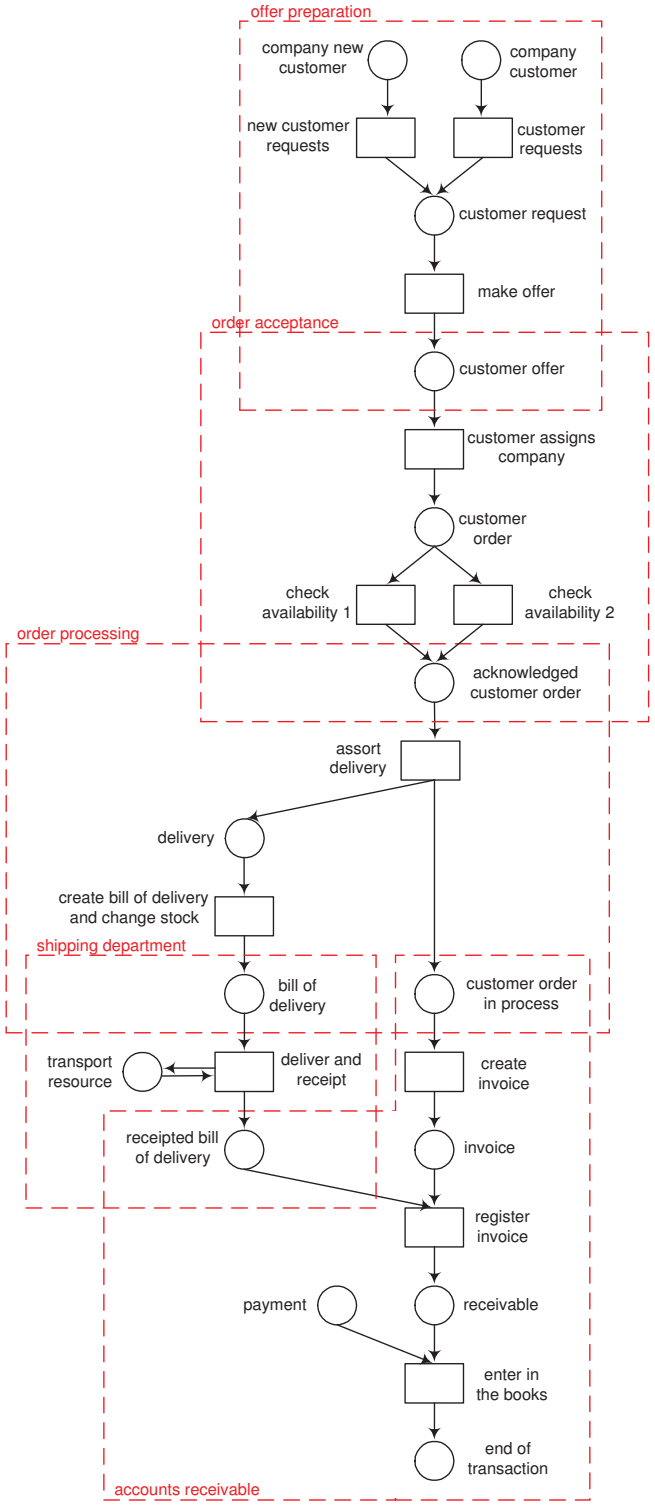



Figure 9.1: P/T-net "Logistics"

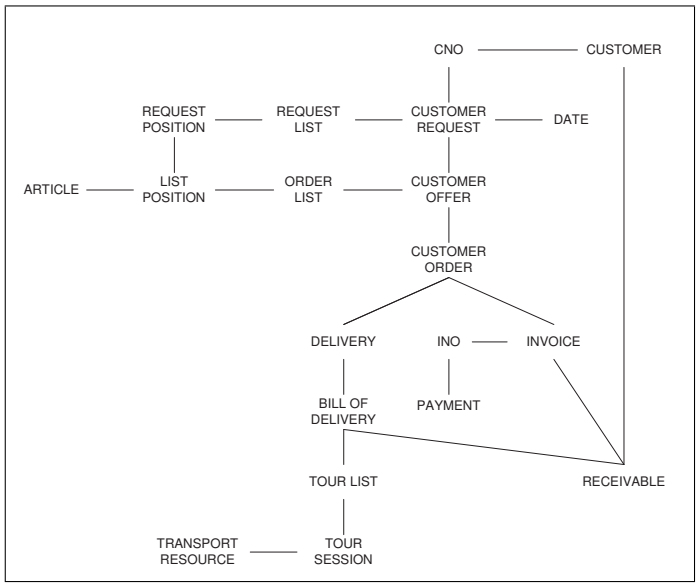



Figure 9.2: Uses-hierarchy of the data type part "Customer"

*offer* and arc inscriptions given by the variable $f$. In the same way we proceed with the net inscription in the environment of the transitions *new customer request* and *customer request* resulting in the AHO-net scheme in the right upper corner in Fig. 9.3, where the net inscriptions besides the predicates *eq* and *and* consist of variables only. We achieve a high-level of abstraction because the operations are not fixed in the net structure but can be given at run time by suitable operations, i.e. tokens on the corresponding places. Moreover, we are able to introduce different kinds of computation, for instance to capture an exception handling by adding further operations.

In a next step we apply the folding construction wrt. product types to reach a more compact description. For instance, the set of places consisting of *P1*, *P1'*, *P1"*, and *P1"'* is uniform, i.e. they have the same transition *new customer request* in their pre- and post domain and the arc inscriptions in the environment of these places are labeled by corresponding variables. Thus these places are replaced by the new place *P1*, where the assigned product type combines the types of these places. The arc inscriptions are obtained by forming the tuple $(f, g, h, k)$. The resulting AHO-net scheme is depicted in the lower row in Fig. 9.3. Note that due to the results in Chapter 5 folding constructions preserve the firing behavior. Hence with respect to an appropriate initial marking the AHO-nets in Fig. 9.3 are semantically equivalent.

In Fig. 9.4 the AHO-net scheme of the department "Order Acceptance" is depicted, while Fig. 9.5 shows the activities of the department concerning the processing of orders. In both cases we apply the folding construction wrt. constant symbols resp. product types several times.

The shipping department is modeled by the AHO-net schemes in Fig. 9.6. Here the transitions *receipt okay*, *receipt rebate* and *receipt refuse* realize three different possibilities for the reception of goods, i.e. the delivery was accepted or all or parts of it were rejected by the customer. The interesting point is that in the AHO-net scheme in the right-lower corner in Fig. 9.6 all these transitions have not only the same places in their environment but are also inscribed by the same net inscriptions. Thus we use the horizontal structuring technique fusion (see Chapter 4) to merge these transitions into one transition *receipt*. For this reason we need suitable AHO-

net scheme morphisms so that the domain of these morphisms consists of an AHO-net scheme with two places *tour list* and *receipted bill* and one transition $t$. The net inscription of this transition is identical to the net inscription of the transitions *receipt okay*, *receipt rebate* and *receipt refuse*. We define two AHO-net scheme morphisms $f_N$ and $g_N$ so that the transition $t$ is mapped by the latter to the transition *receipt okay* and by the former to the transition *receipt rebate*. Then the transitions *receipt okay* and *receipt rebate* are fused into one transition $t'$. In the same way we proceed with the transitions $t'$ and *receipt refuse* and obtain the transition *receipt*.

Finally, the AHO-net schemes modeling the department "Accounts Receivable" are given in Fig. 9.7. An invoice can be handled in different ways depending on the corresponding reception of goods. Besides the normal registration, the invoice may be canceled or partly changed according to the actual delivery. The transitions modeling these different activities (see the AHO-net schemes in the upper-left corner in Fig. 9.7) are fused into one transition (see the AHO-net schemes in the lower-right corner in Fig. 9.7). Now, exceptions of the normal process can be introduced into our system in an elegant way because they do not effect the overall net structure. If there are further possibilities for the reception of goods, this results in a different handling of invoices. In this case, we add some suitable operations as tokens on the corresponding places, while the net structure remains unchanged.

To obtain the overall system of the business process logistics depicted in Fig. 9.8 we first observe that there are places with the same name which are present in different AHO-nets schemes, for instance *customer offer* in Fig. 9.3 and Fig. 9.4. The AHO-net schemes resulting from the application of folding constructions resp. fusions are sequentially composed by merging these places. Formally we employ the horizontal structuring technique union described in Chapter 4, where the interface is given by the overlapping places.

## 9.3 Further Aspects

We have presented a higher-order version of the case study "Logistics", which allows the flexible modeling of the business processes by adding additional operations as tokens. In [Bog04] the case study is first modeled by P/T-nets, which follows the base-line process defined by event driven process chains in [Sch94]. Afterwards the P/T-net is refined by adding further details like transport resources. The resulting P/T-net is depicted in Fig. 9.1. In order to introduce a suitable data structure the model is transformed into an algebraic high-level net. In a first step a trivial data type part is added by using the concept of net class transformations [Urb03]. Then the algebraic high-level net is refined by a more complex specification for the description of the documents and the activities involved in the logistics process. Of course it is often more adequate to apply rules to the low-level version of the case study and to specific components. Here, the results achieved in [Urb03] guarantee not only the compatibility of net class transformations and net model transformations, but also the compatibility of net model transformations and horizontal structuring techniques.

To make a link from the higher-order version of the logistics process presented in this chapter to the low-level and high-level versions it is desirable to have on the one hand suitable net class transformations between the class of algebraic high-level nets and the class of algebraic higher-order nets, and on the other hand the concept of net model transformation for algebraic higher-order nets, so that the compatibility results can be obtained. This will be discussed in more detail in Chapter 10.

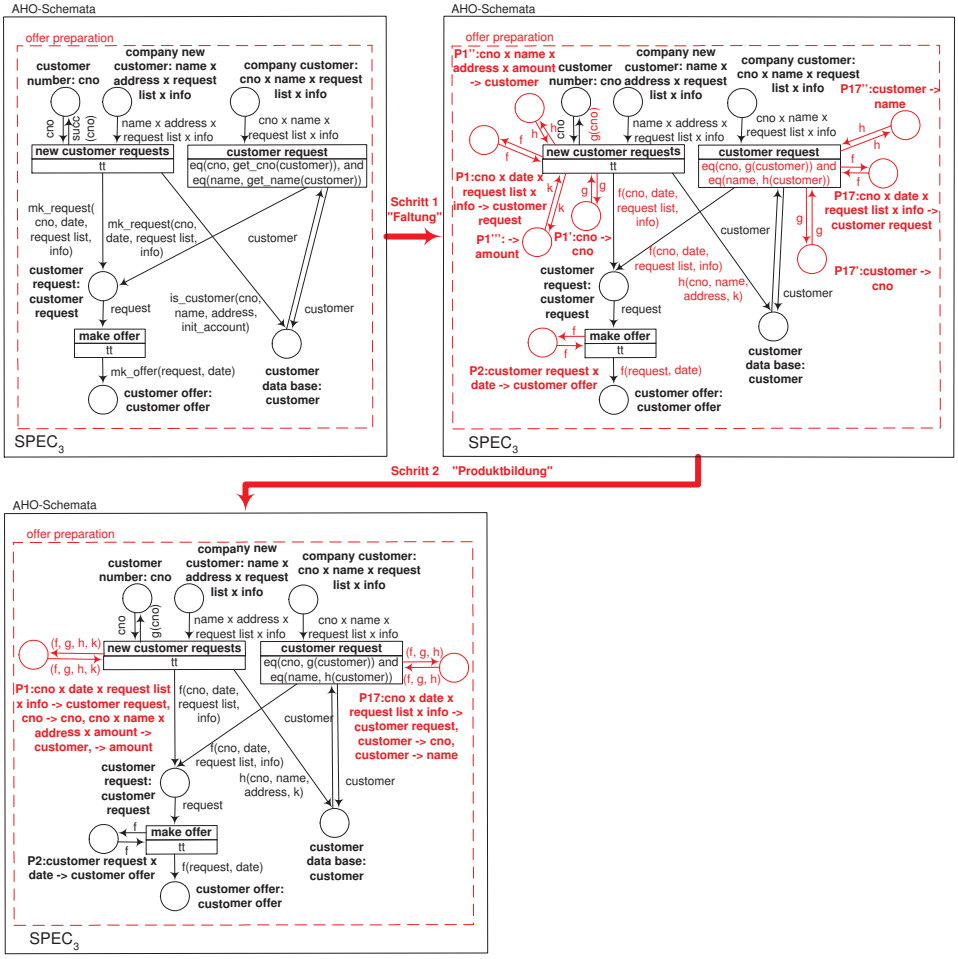



Figure 9.3: AHO-net schemes of department “Offer Preparation”

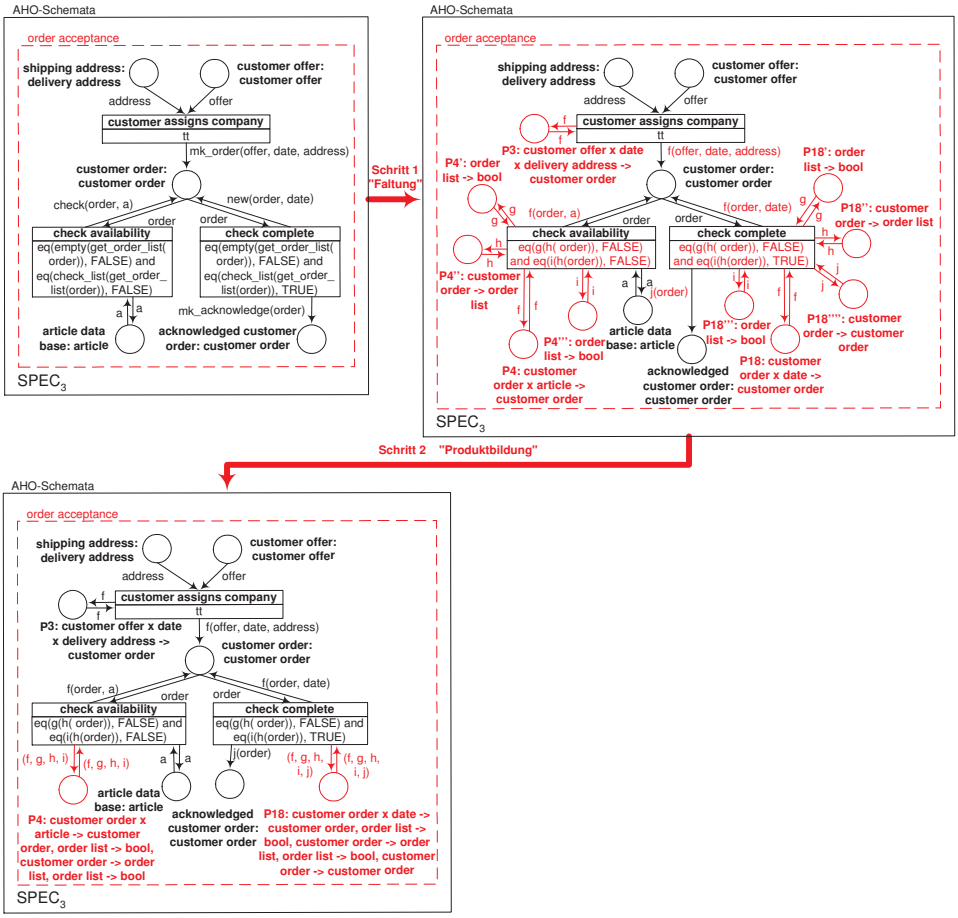



Figure 9.4: AHO-net schemes of department “Order Acceptance”

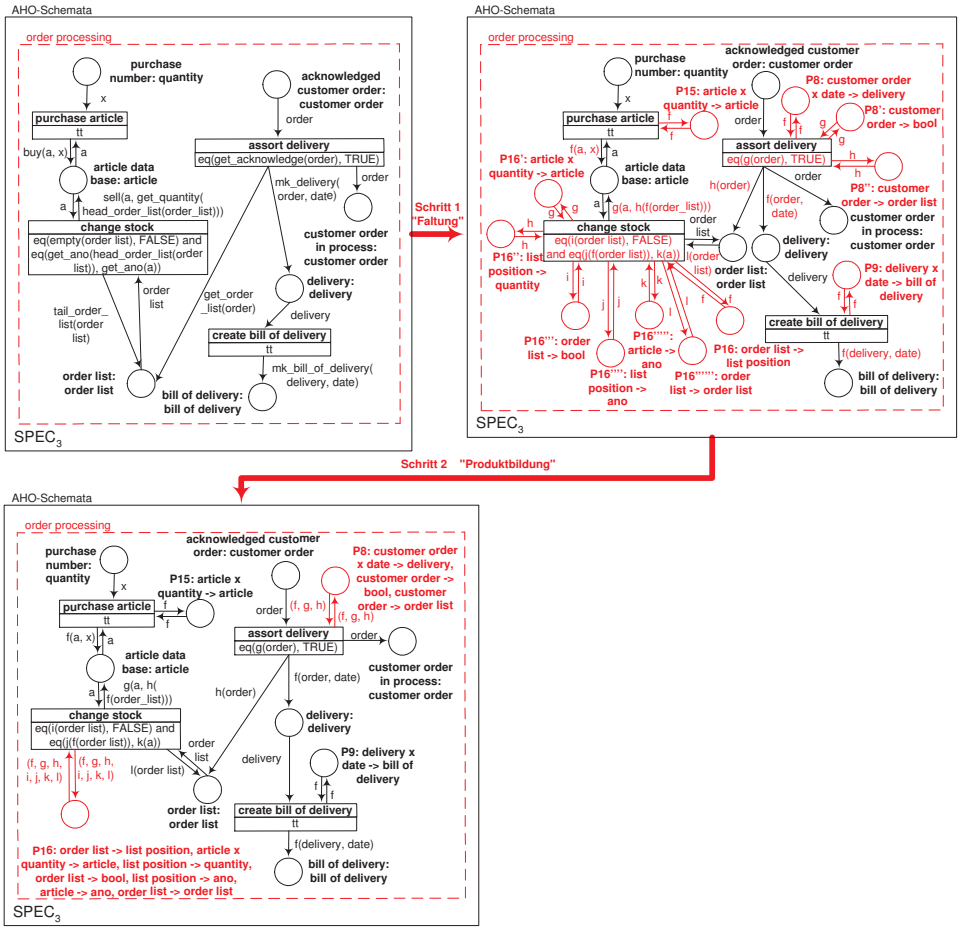



Figure 9.5: AHO-net schemes of department “Order Processing”

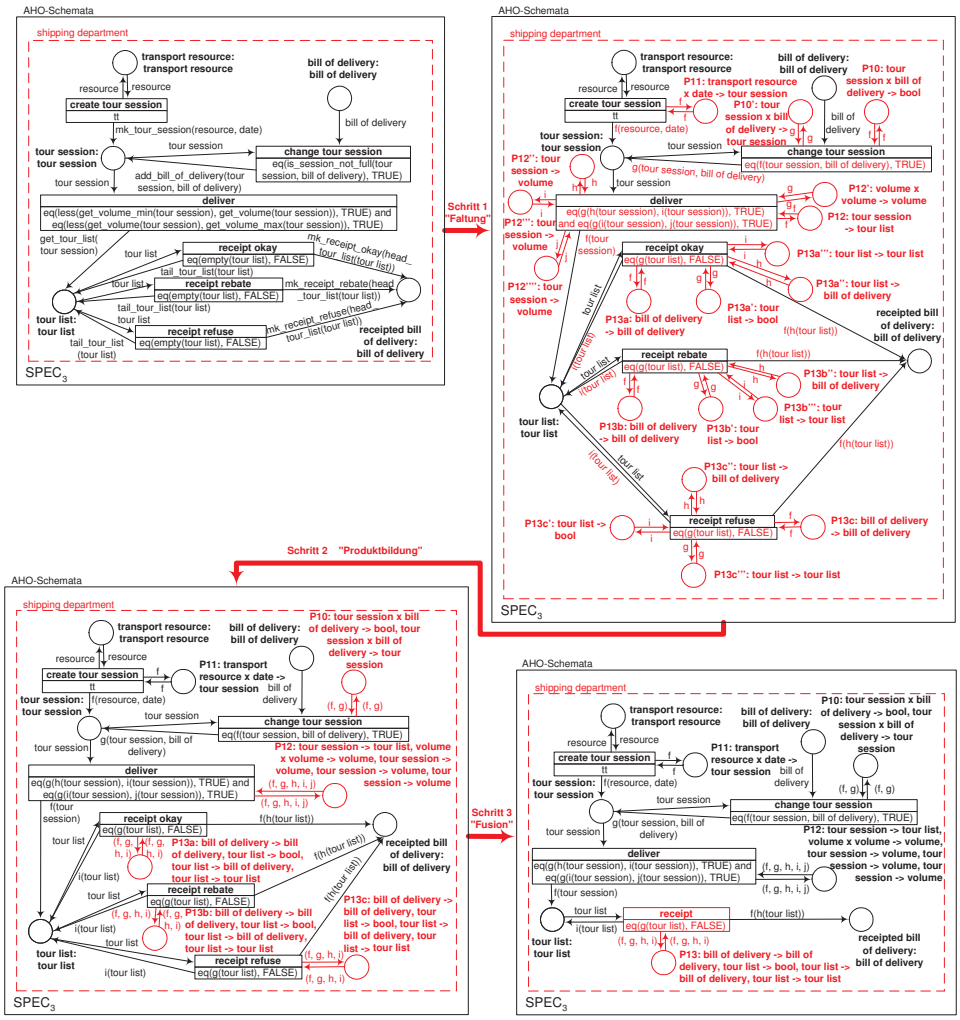



Figure 9.6: AHO-net schemes of department "Shipping"

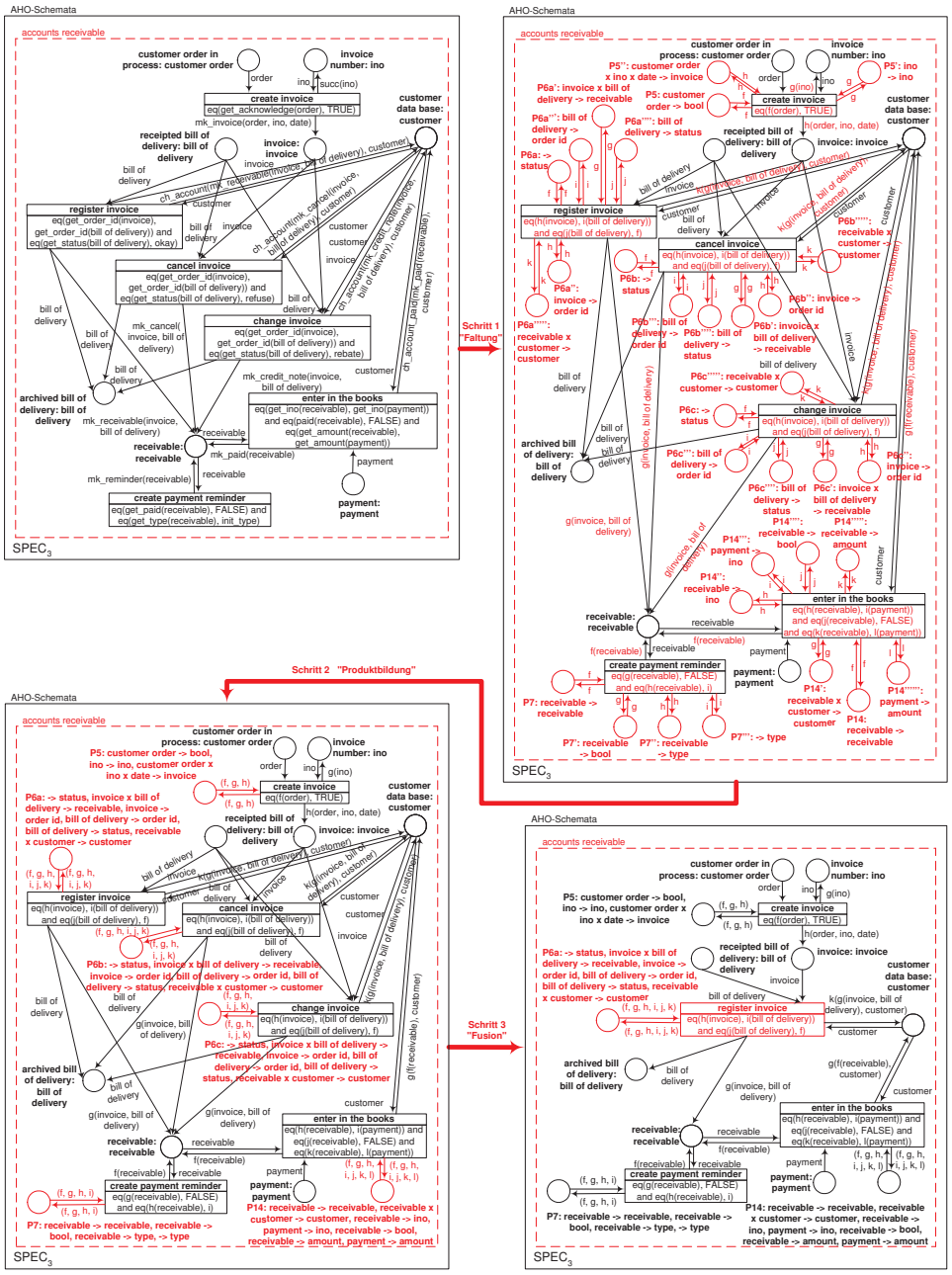



Figure 9.7: AHO-net schemes of department "Receivable"

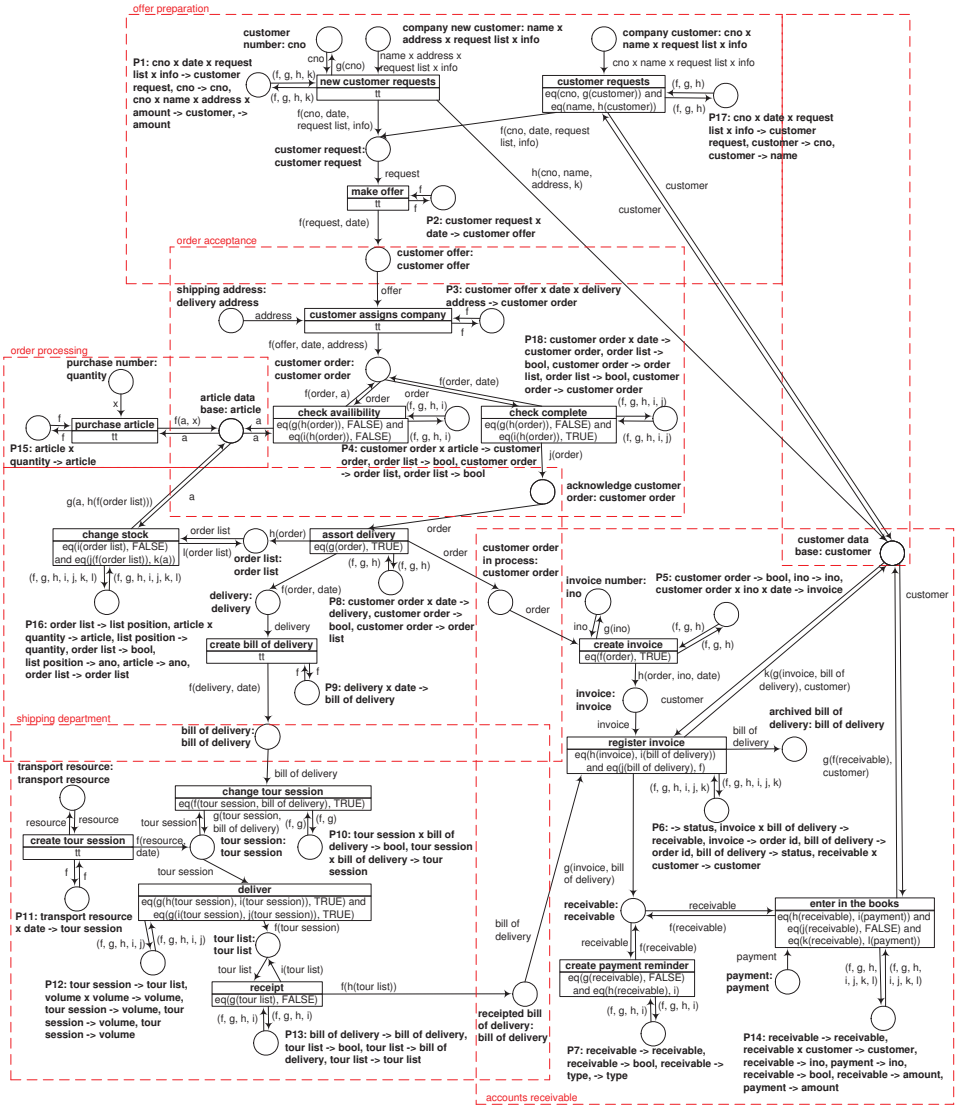



Figure 9.8: AHO-net scheme “Logistics”

# Chapter 10

# Future Work

In this chapter we discuss several interesting problems left for future investigations. Even though we have presented a formal approach for algebraic higher-order nets and have achieved several important results in Chapter 4 and Chapter 5, further investigation and research is necessary to obtain a general theory for algebraic higher-order nets and to facilitate practical applications. The formal approach presented here provides the basis for further theoretical research in three different directions:

- the concept of rule-based transformations and property preserving rules (see Section 10.1) ensuring consistency and compatibility with the horizontal structuring techniques presented in Section 4.1,
- different kinds of net class transformations (see Section 10.2) to establish a link between algebraic higher-order nets and other net classes and to obtain and transfer results achieved for distinguished net classes, and
- the application of folding and unfolding constructions wrt. constant symbols presented (see Section 5.1) and the application of rule-based transformations, so that these activities can be interleaved, leading to the concept of dynamic reconfigurations of processes (see Section 10.3).

Finally, tool support (see Section 10.4) is a main precondition for the practical use of algebraic higher-order nets because not only the data type part but also the net structure might be quite complex, so that users need support not only to specify and simulate, but also to check the correct behavior of their models.

## 10.1 Rule-Based Transformation

The strong relationship between the area of Petri nets and graph transformation systems has been researched in a number of papers. Looking at Petri nets from the perspective of graph grammars, it is quite natural to regard them as grammars acting on discrete graphs. In this way transitions can be represented by graph rules and the application of such a rule simulates the token game (see e.g. [CEL$^+$93, Cor95, KR94]).

The concept of high-level replacement systems [EHKP91] was the starting point to obtain new results for the area of Petri nets. High-level replacement systems provide an abstract framework for rule-based transformations to different domains as for instance place/transition nets [EGP99], algebraic specification [EGP99] and algebraic high-level nets [PER95]. Rules and transformations are given as generalization of these concepts for graph grammars. Especially, the instantiation of

high-level replacement systems to Petri nets leads to the concept of net transformation systems [PER95, Urb03]. The basic idea behind net transformation systems is the stepwise development of systems in the framework of Petri nets based on the double-pushout approach. These transformations are called rule-based transformations, because a high-level replacement system is defined by an arbitrary category and a distinguished class of morphisms used to form rules, i.e. rules in the double-pushout approach are given as a span of two morphisms. Think of these rules as replacement systems, where the left-hand side of the rule is replaced by the right-hand side.

To apply the theory of high-level replacement systems, several conditions, called HLR-conditions, have to be checked. The main advantage is that once these conditions are verified for a specific category, several results are obtained, like local Church Rosser, Parallelism, and Concurrency Theorems. The local Church Rosser Theorem I states that two alternative transformations are concurrent if they are not mutually exclusive, called parallel independence, i.e. the matches do not overlap in items which are deleted by one of these transformations. Symmetrically, the local Church Rosser Theorem II states that two alternative transformations are concurrent, if they can be performed in a different order without changing the result, called sequential independence, i.e. if the match of the second one does not depend on elements generated by the first one, and the second one does not delete an item that has been generated by the first one. The Parallelism Theorem is closely related to the local Church Rosser Theorems. Here, parallel transformations are not described by interleaving but by truly parallel applications and it states necessary and sufficient conditions for the parallelization of two transformations, called synthesis, and for the sequentializing of a parallel transformation, called analysis. Finally, the Concurrency Theorem states under which conditions, called dependency relation, a new rule can be constructed by using two existing rules and vice versa a rule can be split into two rules, which have to be sequentially applied.

To avoid the HLR-conditions the notion of adhesive HLR systems has been introduced in [EHPP04]. In this way mainly the so called van Kampen square property has to be checked instead of the HLR-conditions. Roughly, a van Kampen square is a pushout square which is stable under pullbacks. Moreover, pushouts along a class of distinguished morphisms have to be van Kampen squares. It is shown that most of the results for the theory of high-level replacement systems given above are valid already for adhesive HLR systems, and only the parallelism theorem requires an additional condition. But already P/T-nets fail to be an adhesive HLR system because the construction of free commutative monoids do not preserve pullbacks. Fortunately, most of the results given above can also be formulated under weaker assumptions, where the van Kampen square property has to be stable only under pullbacks, where the morphisms are morphisms of the distinguishable class. This results in the so called weak adhesive HLR systems.

As rule-based transformations were found to be a very important structuring technique for (high-level) Petri nets, it would be desirable to have this notion and the results from above for AHO-nets. As mentioned above this can be realized in two different ways. On the one hand we have to check the HLR-conditions for AHO-nets. On the other hand we have to check whether the data type part of AHO-nets given by higher-order signatures and higher-order algebras is a weak adhesive HLR system. Then we obtain that AHO-nets are a weak adhesive HLR system in a much easier way due to the important result that (weak) adhesive HLR systems are closed under products, functors and comma categories.

To check the properties for both, the data type part and the AHO-nets, can be avoided using the approach of high-level abstract Petri nets [Pad96], which can be instantiated to a great variety of high-level net classes including algebraic high-level nets [PER95, Pad96, EHP$^+$02] and coloured Petri nets [Jen81]. High-level abstract

Petri nets give an abstract notion of high-level net classes by introducing not only for the data type part but also for the net structure part an abstract parameter, which is given for the data type part by the notion of institutions and for the net structure part by a suitable net structure functor. Due to the instantiation we also get the notions of markings, firing behavior, and morphisms for the concrete net class. The advantage is an economic way to achieve the results of cocompleteness of the concrete category and preservation of the firing behavior by morphisms, because these results have been proven on an abstract level and are transferred to the concrete net class. Note that these results are explicitly proven for AHO-net scheme in Section 4.1 and AHO-nets in Section 4.2. But within the framework of high-level abstract Petri nets we also achieve the results of the local Church Rosser and Parallelism Theorems, described above, and the compatibility of rule-based transformations with horizontal structuring techniques.

In detail we mainly have to check if higher-order signatures and algebras are suitable data type parts for the instantiation of high-level abstract Petri nets and if higher-order signatures form a high-level replacement system. While we expect that higher-order signatures form a high-level replacement system, the aspect that higher-order signatures and algebras are of a suitable data type part for the instantiation of high-level abstract Petri nets is more complicated.

The data type part of high-level abstract Petri nets requires a notion of amalgamation, i.e. for every pushout of higher-order signatures there has to be a model of the resulting signature. This model is achieved by the combination of the models of the value signatures wrt. the model of the interface signature. But in the case of higher-order algebras this model can not be uniquely determined because of the mixed function types which are generated by the pushout of higher-order signatures. More precisely, in the resulting higher-order signature we get further function types, which are not a part of the value signatures, due to the union of the basic types. Thus we are free in the interpretation of these function types. A good solution would be to restrict the requirements of the data type part of high-level abstract Petri nets in the sense that amalgamations are not required. The disadvantage of this restriction is that we loose the notion of markings and firing behavior and thereby the result of preservation of the firing behavior by morphisms. However, the results of cocompleteness, local Church Rosser and Parallelism Theorems, and the compatibility of rule-based transformations with horizontal structuring techniques can be achieved because they are formulated for high-level abstract Petri nets without models.

Another problem arises due to the requirement of an initial object in the category of higher-order signature in high-level abstract Petri nets. This initial object is used in the net inscriptions. But in the case of higher-order signatures the initial model is always empty due to the interpretation of operation symbols as partial functions. Thus, the requirements in high-level abstract Petri nets are too restrictive to capture partial algebras and have to be relaxed in the sense that there is a model, which is used for the net inscriptions, but is not necessary the initial object in the corresponding category. Although the fact that the object is initial is frequently used to prove the results given above, these results can, as far as we can see, also be proven with those more relaxed requirements.

Finally, the definition of AHO-nets includes a type function for places, which is not present in the framework of high-level abstract Petri nets. This however, would require an extension of the notion of high-level abstract Petri nets. In detail the data type part of high-level abstract Petri nets has to contain a suitable notion of sorts or types and the net structure functor is a slightly different functor because it has to take the typing function into account. This aspect has been discussed in [EHP+02]. It is still an open question, the results of which can be obtained for this extended version of high-level abstract Petri nets.

Summarizing, we can follow three main directions to obtain the notions and results of rule-based transformations for AHO-nets, high-level replacement systems, adhesive HLR-categories and high-level abstract Petri nets. We would prefer to revise the notion of high-level abstract Petri nets, such that the instantiation leads to the concepts of AHO-nets presented in Section 3.2. In this way much work will be done to obtain results for other high-level net classes, which do not fit into the actual framework of high-level abstract Petri nets.

## 10.2 Net Class Transformation

To establish a link between AHO-nets and other net classes, the so called net class transformations are of particular interest [Urb03]. For AHO-nets there are two main concepts of net class transformations. On the one hand the transformations between AHO-nets and P/T-nets are an important step towards a formal analysis of AHO-nets because in this way we can use the results obtained for place/transition nets and the tools provided for place/transition nets. On the other hand the transformation of AHL-nets into AHO-nets establishes a strong relationship between these net classes and makes the higher-order features and folding constructions available for AHL-nets.

First we discuss a net class transformation which reflects the independence of the net structure component and the data type component of AHO-nets. The interpretation of the net structure component as a place/transition net represents all processes in the AHO-net if we abstract from the data elements, which are calculated by the firing of transitions. This type of net class transformation makes the standard theory and analysis methods for place/transition nets applicable to our AHO-nets. The transformation of an AHO-net into a place/transition net can be easily achieved by forgetting the data type part, the terms in the net inscriptions, and the typing of places, and results in a so-called skeleton of AHO-nets. Vice versa the transformation of a place/transition net into an AHO-net adds a simple data type part consisting of only one basic type *token* for black tokens. Moreover, the net inscriptions in the resulting AHO-net are decorated by variables of basic type *token* instead of natural numbers. We expect that these constructions can be turned into functors which preserve pushouts and initial objects.

The transformation between the net class of AHL-nets and the net class of AHO-nets is more advanced, because here we have to transform the classical data type part into a higher-order data type part. Problems arise due to the partiality of higher-order models, because classical models are in a total setting. Moreover, the firing conditions are given in AHO-nets by atomic formulas, while we have a set of equations in AHL-nets. Thus, in the following we first describe the transformation of an AHL-net scheme, i.e. an AHL-net without a classical algebra, into an AHO-net scheme, and discuss a possible solution of the problems given above afterwards.

The canonical embedding of classical signatures into higher-order signatures is based on the observation that CASL can be embedded into HASCASL via so called institution comorphisms. Given a classical signature, the set of basic types in the corresponding higher-order signature consists of the same sorts as in the classical signature, while the operations of the classical signature are turned into constants of appropriate higher-order type. As mentioned above in AHL-nets, the firing conditions are given by a set of equations, while in AHO-nets the firing conditions are defined by atomic formulas. Thus, we introduce some predicates to capture the firing conditions of an AHL-net scheme in the corresponding AHO-net scheme, i.e. we extend the higher-order signature by one predicate $eq_s : Pred(s \star s)$ for each basic type $s \in S$ and a predicate $\& : Pred(Pred(unit) \star Pred(unit))$ for the combination of atomic formulas. Furthermore, we need a type of boolean values, i.e. we introduce

predicates *tt* and *ff* of type *Pred*(*unit*).

In the following we discuss the transformation of AHL-net schemes into AHO-net schemes. Given an AHL-net scheme $N = (\Sigma, P, T, pre, post, cond, type)$ with a classical signature $\Sigma$ instead of an specification. The corresponding AHO-net scheme *HO-N* consists of the same sets of places $P$ and transitions $T$. The data type part of *HO-N* is given by the corresponding higher-order signature *HO*-$\Sigma$ as described above. While the typing of the places remains unchanged, we have to transform the terms in the net inscriptions into higher-order terms, i.e. each operation symbol in the net inscriptions is replaced by the corresponding constant symbol of appropriate type. Finally we have to transform the firing conditions. Let $cond(t) = \{e_1, \ldots, e_n\}, n \geq 0$, for some $t \in T$. Then for $n = 0$ we have $cond_N(t) = \emptyset$, which is transformed into $cond_{HO\text{-}N}(t) = tt$. For $n \geq 1$ each equation $(term_1 = term_2) \in cond_N(t)$ with terms $term_1$ and $term_2$ is transformed into an atomic formula $eq_s(term'_1, term'_2)$, where $term'_1$ and $term'_2$ are the higher-order terms resulting by the replacement of operation symbols by corresponding constant symbols of appropriate type. If there are more than one equation, the resulting atomic formulas are composed into one atomic formula by using the predicate &.

We are convinced that the construction described above can be turned into a functor and we expect that this functor is $\mathcal{M}$-compatible (see [Urb03]) in the sense that the net class transformation of AHL-net schemes into AHO-net schemes is compatible with rule-based transformations as described in Section 10.1.

For the level of semantics the extension of a classical algebra to a higher-order partial algebra can not be uniquely obtained because there are at least two different ways of extension: the free extension, where function types are in a sense minimal, and a standard extension, where function types are interpreted by the full function space. Moreover, the problem of partiality arises, because classical algebras are in a total setting while we have higher-order partial algebras. One solution might be to handle partiality by using an error mechanism on the level of classical algebras. But it is an open question, which translation of classical algebras into higher-order partial algebra is suitable for our purpose.

Moreover, in the category of AHL-nets we are using generalized morphisms to modify both parts of the data type part, the specification and the algebra. But AHO-net morphisms are restricted in the sense that they only modify the higher-order partial algebra, while the higher-order signature is fixed. To establish a relation between these net classes we need a kind of generalized morphism for AHO-nets first, which is part of future work.

## 10.3 Dynamic Reconfiguration

An interesting aspect for future work is the combination of rule-based transformations and the folding and unfolding constructions wrt. constant symbols presented in Section 5.1, leading to the concept of dynamic reconfigurations of processes. In this section we discuss this kind of dynamic reconfiguration on the basis of a small example inspired by our case study of the medical information system (see Chapter 8). In Fig. 10.1 an AHO-net is depicted, the subsystem of initializing the patient record and taking of vital values. The AHO-net is decorated with terms over a higher-order signature, which is not given here. The idea of the AHO-net is to model the following situation in a hospital: The patient is located in the ward. His/her PadID is used to initialize his/her patient record containing e.g. all measured values. Afterwards a single vital value can be taken e.g. for the diagnosis and therapy by a doctor if this treatment is demanded in the patient record.

This model is useful in the area of processes, which are fixed once and for all and where no further changes are required. But we have already mentioned, medical

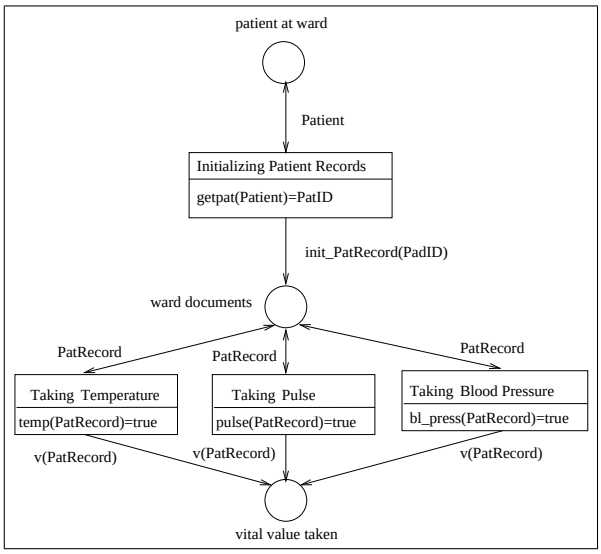



Figure 10.1: Algebraic higher-order net "Patient Record"

processes have to be often reorganized, e.g. the initialization of the patient record has to be modified. In this case the relevant parts of the model have to be replaced by the modified process. Here we can use the folding construction wrt. constant symbols to obtain a more flexible model. For a detailed explanation we refer to Section 5.1. As a result of the folding construction wrt. constant symbols we obtain the AHO-net which is shown on the left hand side (the non bold face part) of Fig. 10.2. Because folding constructions preserve the firing behavior, the folded net models the original medical-process.

The main idea of dynamic reconfiguration is the refinement of the transition *Initializing Patient Records* by a sequential process, where the temperature curve becomes initialized first and the therapy prescription or the medicine prescription afterwards. Thus, we introduce the new process, corresponding to the refinement of the transition *Initializing Patient Records* described above, into our model by using rule-based transformations, so that we obtain the AHO-net in Fig. 10.2. There are further transitions to switch into the new process. The transition *Changing Initialization* realizes the replacement of the token *init Record* by the tokens *init TempC*, *init ThPres* and *init MedPres*.

In Fig. 10.2 the new process is still inactive. But if the transition *Changing Initialization* fires, the new process is active and the old one is inactive. Then there is no follower marking so that the transition *Initializing Patient Records* and the transition *Changing Initialization* are enabled. Thus, these transitions and all arcs in their environment can be deleted at a certain point of time. If necessary we can unfold the resulting AHO-net because the firing behavior is preserved by the unfolding construction.

In the following we discuss the general concept of dynamic reconfiguration where the AHO-net in Fig. 10.2 is only a specific example. We assume that a formal model given by an AHO-net $N$ exists. If modifications of subprocesses are required, e.g. the refinement of a single step into a sequential process, we identify the relevant subprocess $N_1$, which is a part of the existing model $N$. Next we fold the subprocess $N_1$ into a AHO-net using an appropriate folding construction wrt. constant symbols and get a subprocess $F(N_1)$. Because the folding construction preserves the firing behavior, both, the original model $N$ and the folded model $N_C = N - N_1 + F(N_1)$, are semantically equivalent. Next we assume that the modified process is given by

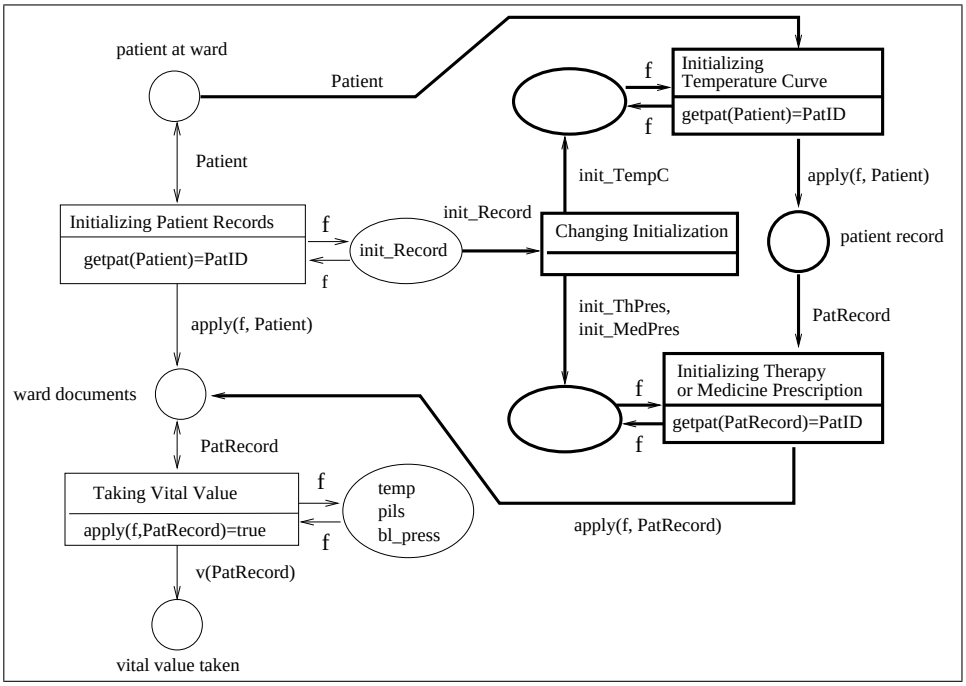



Figure 10.2: Algebraic higher-order net "Patient Record II"

a AHO-net $N_2$, so that the marking of contextual places is empty.

Now the net $N_C$ and the subprocess $N_2$ are composed using the structuring technique union with an interface net containing the relevant places. Furthermore we add a transition $t_{change}$. Its firing behavior realizes the switching into the new subprocess $N_2$. In terms of Petri nets, on the one hand the pre domain of the transition $t_{change}$ is constructed using the contextual places of the process $F(N_1)$ and arcs labeled with constants, corresponding to the marking of the contextual places. On the other hand the post domain of the transition $t_{change}$ is constructed using the contextual places of the modified process $N_2$ and arcs labeled with constants, corresponding to the marking of contextual places, which starts the modified process $N_2$.

In the resulting AHO-net $N_{HO}$ the modified process $N_2$ is inactive. At a certain point of time the transition $t_{change}$ is enabled and fires, so that as a result the modified process $N_2$ is active and the process $N_1$ is inactive. Because the transitions in the subprocess $N_1$ and the transition $t_{change}$ are not enabled under the follower markings, they can be deleted together with their environment by an appropriate rule. Now the resulting AHO-net $N'_C$ can be unfolded into an AHO-net $U(N'_C)$.

Note that a firing step of a transition and both the modification of a transition and the folding and unfolding construction are in conflict. In this way subprocesses, which must be modified, are locally transformed, while the global process is not interrupted. Thus, these activities are interleaved. This problem is closely related to basic higher-order net and rule systems (see Remark 6.2.5), where a P/T-system and a set of token rules are considered as reconfigurable P/T-system. Analogously, it would be especially important to analyze under which conditions the token game activities are independent of both the folding and unfolding constructions and rule-based transformations. Here we can base it on the local Church-Rosser properties obtained for rule-based transformations (see Section 10.1).

## 10.4 Tool Support

A concrete implementation of AHO-nets has to be worked out in order to evaluate and improve their applicability. Major problems, which could be solved by such an implementation are:

- a clean interface between the net structure component and the data type part component; especially due to the higher-order signature dynamic tokens with a rather complex structure are imported into the nets,
- a solid and flexible process structure for the concurrent firing of transitions, and
- a general but flexible enough architecture of the whole system, which supports distributed computing, modern interface for user interaction, efficient data storage, security and safety issues and dynamic configuration.

We would like to make extensive use of available tools for (high-level) nets. Unfortunately, this is not possible using tools [RWL$^+$03] for coloured Petri nets [Jen92]. Actually, coloured Petri nets are based on an extension of the functional language Standard ML [MTHM97]. As Standard ML does not allow functional equivalence testing, it is not suitable for our purpose where we need a form of functional equivalence.

However, several aspects of AHO-nets are supported by tools. The algebraic approach to graph transformations, which can also be used for rule-based transformations of nets, is supported by the graph transformation environment AGG (see the homepage of [AGG]). AGG includes an editor for graphs and graph grammars, a graph transformation engine, and a tool for the analysis of graph transformations. On top of the graph transformation system AGG is the GenGED environment (see the homepage of [Gen]), which supports the generic description of visual modeling languages for the generation of graphical editors and the simulation of the behavior of visual models. Especially rule-based transformations for P/T-systems can be expressed using GenGED. These transformations can be coupled to other Petri net tools using the Petri Net Kernel [KW01], a tool infrastructure for editing, simulating and analyzing Petri nets of different net classes and for integration of other Petri net tools. Tools for HasCasl have been already implemented [Mos05], which is an important step towards implementation and tool support for AHO-nets. The Heterogeneous Tool Set (Hets) (see the homepage of [Het]) provides a parser and static analysis for Casl and HasCasl-specifications; theorem proving support in form of a translation to the Isabelle/HOL prover is under development. Also, a translation tool from a HasCasl subset to Haskell is provided.

# Chapter 11

# Conclusion

We conclude with a summary of achieved notions and results, while a discussion of open problems has been given in the previous chapter. In this thesis we have presented the new high-level net classes of algebraic higher-order net schemes and algebraic higher-order nets (see Chapter 3) including higher-order features, product types as well as partiality. We have achieved this by using as data type part higher-order partial algebras in an intensional setting (see Section 3.1), where function types are interpreted by arbitrary sets with an application operation of appropriate type. Thus we are able to give a set theoretic definition of domains and operations and staying close to classical algebraic specifications (see Section 2.2). We have covered the basic notions of algebraic higher-order net schemes as well as algebraic higher-order nets like net structure and operational behavior (see Chapter 3). Moreover we have explored algebraic higher-order net schemes and algebraic higher-order nets on a categorical level by introducing the notions of algebraic higher-order net scheme morphisms as well as algebraic higher-order net morphisms (see Chapter 4). In the case of algebraic higher-order net morphisms we have shown that morphisms are compatible with the operational behavior (see Thm. 4.2.3).

In particular we have achieved several results, so that the following requirements of the introduction are fulfilled as stated below.

**Composition of Models** The horizontal structuring technique called union, which allows the formal construction of larger models from model segments with shared subparts, was achieved by the existence of all pushouts in the category of algebraic higher-order net schemes (see Thm. 4.1.3).

**Adaptive Models** The adaptability of models was tackled in two different directions. On the one hand the variable part, which depends on the current execution, could be as large as required because in our approach of higher-order algebras functions are first-class citizens and the order of functions is allowed to be arbitrarily high. In this way functions can be used as tokens to achieve an operation late-binding mechanism, i.e. depending on a specific application the current execution of the process and reaction to feedback can be given at run-time by adding and exchanging suitable operations as tokens. On the other hand we have introduced folding constructions wrt. constant symbols to support the development of abstract and flexible models. By applying these constructions the level of abstraction increases because certain operations move from the fixed part given by the net structure, into the variable part expressed by tokens. We have shown, that these constructions preserve the operational behavior (see Thm. 5.1.9). These folding constructions were frequently used in our case study "Logistics" (see Chapter 9). Moreover we have introduced an unfolding constructions wrt. constant symbols, which

preserves the operational behavior (see Thm. 5.1.9), so that the folding and unfolding construction wrt. constant symbols are inverse to each other (see Thm. 5.1.6).

**Compact Description of Models** To obtain compact descriptions of models we have achieved two main results. On the one hand we have presented folding constructions wrt. product types, which under certain conditions transfer specific parts of the process specification into the data type part and preserve the firing behavior (see Thm. 5.2.12). These constructions were frequently used in our case study "Logistics" (see Chapter 9). Moreover we have presented unfolding constructions wrt. product types, which preserve the operational behavior (see Thm. 5.2.12), so that the folding and unfolding construction wrt. product symbols are inverse to each other (see Thm. 5.2.7). On the other hand the horizontal structuring technique to identify distinguished elements, called fusion, was achieved due to the cocompleteness of the category of algebraic higher-order net schemes (see Thm. 4.1.4).

**Dynamic Tokens** Here we would like to point out once more that in our approach functions are first-class citizens and the order of functions is allowed to be arbitrarily high. Thus our approach is powerful enough to consider dynamical tokens in the system, having their own individual behavior like Petri nets, as tokens themselves (see Section 6.1, Section 6.2, and Chapter 8).

**Rule Tokens** In Section 6.2 and our case study of medical information systems in Chapter 8 we have demonstrated the use of rules as tokens and transformations as appropriate operations, which are covered by the data type part of higher-order algebras. This leads to a high level of abstraction, which is flexible in the sense that different transformations are not attached to the net structure in a fixed way, but could be expressed by a set of rule tokens. Furthermore, changes of specific transformations result in an exchange of rule tokens. In Section 6.3 we went one step further. Here we have used rules as tokens not only to represent mobile policies, but also for the manipulation of rules themselves.

Furthermore, we have provided the basis not only for transformations and dynamic reconfiguration of models (see Chapter 10) but also for the integration of software components. Although the concepts of rule-based transformations are in general available for algebraic higher-order nets, further research is necessary to obtain the compatibility results for our approach. Another promising topic is the formalization of dynamic reconfiguration, which is not straightforward. Because the order of functions is allowed to be arbitrarily high in our approach, it is possible to make a considerable step forward and reflect specific parts of software components as functions, so that local modification and/or substitution of these parts could be expressed by exchanging the relevant tokens.

Summarizing, we have introduced the novel approach of algebraic higher-order nets in this thesis, where the data type part is extended to include function types, product types, and partiality, so that some major inherent problems of high-level nets like operation late-binding mechanisms are solved. Thus, we obtain a high-level of abstraction in our approach and support structure flexibility and system adaptability in an extensive way. We have presented several large examples and two case studies to demonstrate the practical use of algebraic higher-order nets. We are convinced that this thesis is a good starting point for the future work as discussed in Chapter 10.

# Bibliography


[AC92] E. Astesiano and M. Cerioli. Partial higher-order specifications. *Fundam. Inform.*, 16(1):101–126, 1992.

[ADCR01] G. Agha, F. De Cindio, and G. Rozenberg, editors. *Concurrent Object-Oriented Programming and Petri Nets*, LNCS 2001. Springer, 2001.

[AGG] AGG Homepage. http://tfs.cs.tu-berlin.de/agg.

[AHS90] J. Adamek, H. Herrlich, and G. Strecker. *Abstract and Concrete Categories*. Series in Pure and Applied Mathematics. John Wiley and Sons, 1990.

[Bau90] B. Baumgarten. *Petrinetze, Grundlagen und Anwendungen*. BI Wissenschaftsverlag, 1990.

[BBPP04] M. A. Bednarczyk, L. Bernardinello, W. Pawlowski, and L. Pomello. Modelling mobility with Petri hypernets. In J.L. Fiadeiro, P.D. Mosses, and F. Orejas, editors, *17th Workshop on Recent Trends in Algebraic Development Techniques*, LNCS 3423, pages 28–44. Springer, 2004.

[BFA99] E. Bertino, E. Ferrari, and V. Atluri. The Specification and Enforcement of Authorization Constraints in Workflow Management Systems. *ACM Trans. Inf. Syst. Secur.*, 2(1):65–104, 1999.

[Bog04] J. Bogen. Schrittweise Entwicklung von Ereignisgesteuerten Prozessketten zu Algebraischen Higher Order Netzen. Master's thesis, Technical University Berlin, 2004.

[BW90] M. Barr and C. Wells. *Category theory for computing science*. Prentice-Hall International, 1990.

[CEL+93] Andrea Corradini, Hartmut Ehrig, Michael Löwe, Ugo Montanari, and Francesca Rossi. Abstract Graph Derivations in the Double Pushout Approach. In Hans Jürgen Schneider and Hartmut Ehrig, editors, *Dagstuhl Seminar on Graph Transformations in Computer Science*, LNCS 776, pages 86–103. Springer, 1993.

[CFJF02] S. Chapin, B. D. Faatz, S. Jajodia, and A. Fayad. Consistent policy enforcement in distributed systems using mobile policies. *Data Knowl. Eng.*, 43(3):261–280, 2002.

[CGRW95] I. Claßen, M. Große-Rhode, and U. Wolter. Categorical Concepts for Parameterized Partial Specifications. *Mathematical Structures in Computer Science*, 5(2):153–188, 1995.

[Cor95] A. Corradini. Concurrent computing: from Petri nets to graph grammars. *Electr. Notes Theor. Comput. Sci.*, 2, 1995.

[DFJM00] V. Doshi, A. Fayad, S. Jajodia, and R. MacLean. Using attribute certificates with mobile policies in electronic-commerce applications. In *16th Annual Computer Security Applications Conference*, pages 298–307. IEEE Computer Society, 2000.

[DG91] W. Deiters and V. Gruhn. Software Process Model Analysis Based on FUNSOFT Nets. *Systems Analysis Modelling Simulation*, 8(4-5):315–325, 1991.

[DG94] W. Deiters and V. Gruhn. The FunSoft Net Approach to Software Process Management. *International Journal on Software Engineering and Knowledge Engineering*, 4(2):229–256, 1994.

[EGP99] H. Ehrig, M. Gajewsky, and F. Parisi-Presicce. High-Level Replacement Systems with Applications to Algebraic Specifications and Petri Nets. In G. Rozenberg, U. Montanari, H. Ehrig, and H.-J. Kreowski, editors, *Handbook of Graph Grammars and Computing by Graph Transformations, Volume 3: Concurrency, Parallelism, and Distribution*, chapter 6, pages 341–400. World Scientific, 1999.

[EHKP91] H. Ehrig, A. Habel, H.-J. Kreowski, and F. Parisi-Presicce. Parallelism and Concurrency in High-Level Replacement Systems. *Math. Struct. in Comp. Science*, 1:361–404, 1991.

[EHP+02] H. Ehrig, K. Hoffmann, J. Padberg, P. Baldan, and R. Heckel. High-Level Net Processes. In W. Brauer, H. Ehrig, J. Karhumäki, and A. Salomaa, editors, *Formal and Natural Computing*, LNCS 2300, pages 191 – 219. Springer, 2002.

[EHPP04] H. Ehrig, A. Habel, J. Padberg, and U. Prange. Adhesive High-Level Replacement Categories and Systems. In F. Parisi-Presicce, P. Bottoni, and G. Engels, editors, *2nd Int. Conference on Graph Transformation*, LNCS 3256, pages 144–160. Springer, 2004.

[Ehr79] H. Ehrig. Introduction to the Algebraic Theory of Graph Grammars (A Survey). In *Graph Grammars and their Application to Computer Science and Biology*, LNCS 73, pages 1–69. Springer, 1979.

[EM85] H. Ehrig and B. Mahr. *Fundamentals of Algebraic Specification 1: Equations and Initial Semantics*, volume 6 of *EATCS Monographs on Theoretical Computer Science*. Springer, Berlin, 1985.

[EP91] H. Ehrig and F. Parisi-Presicce. Nonequivalence of Categories for Equational Algebraic Specifications in View of High-Level Replacement Systems. Technical Report 91-16, Technical University Berlin, 1991. Short version in Proc. 3rd Conf. on Algebraic and Logic Programming, Pisa, 1992.

[EP04] E. Ehrig and J. Padberg. Graph Grammars and Petri Net Transformations. In *Lectures on Concurrency and Petri Nets. Special Issue Advanced Course PNT*, LNCS 3098, pages 496–536. Springer, 2004.

[Erm96] C. Ermel. Anforderungsanalyse eines medizinischen Informationssystems mit Algebraischen High-Level-Netzen. Technical Report 96-15, TU Berlin, 1996.

[Far99] B. Farwer. A Linear Logic View of Object Petri Nets. *Fundam. Inform.*, 37(3):225–246, 1999.

[Far00] B. Farwer. A Multi-region Linear Logic Based Calculus for Dynamic Petri Net Structures. *Fundam. Inform.*, 43(1-4):61–79, 2000.

[Fef92] S. Feferman. A New Approach to Abstract Data Types, I: Informal Development. *Mathematical Structures in Computer Science*, 2(2):193–229, 1992.

[GB84] J. A. Goguen and R. M. Burstall. Introducing Institutions. In *Carnegie Mellon Workshop on Logic of Programs*, pages 221–256. Springer, 1984.

[Gen] GenGED Homepage. http://tfs.cs.tu-berlin.de/genged.

[Gen86] Hartmann J. Genrich. Predicate/Transition Nets. In Wilfried Brauer, Wolfgang Reisig, and Grzegorz Rozenberg, editors, *Advances in Petri Nets*, LNCS 254, pages 207–247. Springer, 1986.

[GL81] H.J. Genrich and K. Lautenbach. System Modelling with High-Level Petri Nets. *Theoretical Computer Science*, 13:109–136, 1981.

[God83] H.P. Godbersen. *Funktionsnetze*. Ladewig-Verlag, 1983.

[Gru91] V. Gruhn. *Validation and Verification of Software Process Models*. PhD thesis, Universität Dortmund, Abteilung Informatik, 1991.

[GTW78] J. A. Goguen, J. W. Thatcher, and E. G. Wagner. Abstract Data Types as Initial Algebras and the Correctness of Data Representations. In R. Yeh, editor, *Current Trends in Programming Methodology*, volume 4, pages 80–149. Prentice-Hall, 1978.

[Han97] Yanbo Han. *Software Infrastructure for Configurable Workflow System - A Model-Driven Approach Based on Higher-Order Nets and CORBA*. PhD thesis, Technical University of Berlin, 1997.

[HEM05] K. Hoffmann, H. Ehrig, and T. Mossakowski. High-Level Nets with Nets and Rules as Tokens. In *26th International Conference on Application and Theory of Petri Nets,*, LNCS 3536, pages 268–288. Springer, 2005.

[Het] Hets Homepage. http://www.tzi.de/cofi/hets.

[HM02] K. Hoffmann and T. Mossakowski. Algebraic Higher-Order Nets: Graphs and Petri Nets as Tokens. In *16th International Workshop on Recent Trends in Algebraic Development Techniques*, LNCS 2755, pages 253–267. Springer, 2002.

[HMPP04] K. Hoffmann, Till Mossakowski, and F. Parisi-Presicce. Higher-Order Nets for Mobile Policies. In G. Rozenberg, H. Ehrig, and J. Padberg, editors, *Workshop on Petri Nets and Graph Transformations, Satellite Event of ICGT'04*, volume 127 of *Electr. Notes Theor. Comput. Sci.*, pages 87–105, 2004.

[Hof00] K. Hoffmann. Run Time Modification of Algebraic High Level Nets and Algebraic Higher Order Nets using Folding and Unfolding Constructions. In G. Hommel, editor, *3th Int. Workshop of Communication Based Systems*, pages 55–72. Kluwer, 2000.

[Hof03] K. Hoffmann. Case Study Logistics: Flexible Modeling of Business Processes using Algebraic Higher-Order Nets. In H. Ehrig, W. Reisig, G. Rozenberg, and H. Weber, editors, *Advances in Petri Nets: Petri Net Technology for Communication Based Systems*, LNCS 2472, pages 145–160. Springer, 2003.

[Hum89] U. Hummert. *Algebraische High-Level Netze*. PhD thesis, Technische Universität Berlin, 1989.

[Jen81] K. Jensen. Coloured Petri Nets and the Invariant Method. *Theoretical Computer Science*, 14:317–336, 1981.

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

[Jen95] K. Jensen. *Coloured Petri Nets - Basic Concepts, Analysis Methods and Practical Use*, volume 2: Analysis Methods. Springer Verlag, EATCS Monographs in Theoretical Computer Science edition, 1995.

[Jen97] K. Jensen. *Coloured Petri Nets - Basic Concepts, Analysis Methods and Practical Use*, volume 3: Practical Use. Springer Verlag, EATCS Monographs in Theoretical Computer Science edition, 1997.

[JV87] E. Jessen and R. Valk. *Rechensysteme: Grundlagen der Modellbildung*. Studienreihe Informatik. Springer, Berlin, 1987.

[Kie04] Ch. Kiesner. Modellierung eines medizinischen Informationssystems mit Algebraischen Higher-Order Netzen. Master's thesis, Technical University Berlin, 2004.

[KPP02] M. Koch and F. Parisi-Presicce. Describing Policies with Graph Constraints and Rules. In A. Corradini, H. Ehrig, H.-J. Kreowski, and G. Rozenberg, editors, *First International Conference on Graph Transformation*, LNCS 2505, pages 223–238. Springer, 2002.

[KR94] M. Korff and L. Ribeiro. An attributed graph transformation approach to the behaviour of algebraic high-level nets. Extended abstract for the international Workshop on Graph Grammars'94, 1994.

[Krä89] B. Krämer. *Concepts, Syntax, and Semantics of SEGRAS*. PhD thesis, Technical University Berlin, 1989. GMD-Bericht Nr. 179.

[Kum01] Olaf Kummer. Introduction to Petri Nets and Reference Nets. *Sozionik Aktuell*, 1:1–9, 2001. ISSN 1617-2477.

[Kum02] Olaf Kummer. *Referenznetze*. Logos Verlag, Berlin, 2002.

[KW01] E. Kindler and M. Weber. The petri net kernel – an infrastructure for building petri net tools. *Software Tools for Technology Transfer*, 3(4):486–497, 2001.

[LH94] M. Löwe and Y. Han. Process Modelling and Control with Higher-Order Nets. Technical Report TR 94-34, Technical University Berlin, 1994.

[Lib] CASL and HASCASL Libraries. http://www.cofi.info/Libraries/.

[Lil95] J. Lilius. *On the Structure of High-Level Nets*. PhD thesis, Helsinki University of Technology, 1995. Digital Systems Laboratory, Research Report 33.

[LWH95] M. Löwe, D. Wikarski, and Y. Han. Higher-Order Object Nets and Their Application to Workflow Modeling. Technical Report TR 95-34, Technical University Berlin, 1995.

[MM90] J. Meseguer and U. Montanari. Petri Nets are Monoids. *Information and Computation*, 88(2):105–155, 1990.

[MOM89] N. Martí-Oliet and J. Meseguer. From Petri Nets to Linear Logic. In *Category Theory and Computer Science*, LNCS 389, pages 313–340. Springer, 1989.

[Mos04] P. D. Mosses, editor. *CASL Reference Manual - The Complete Documentation of the Common Algebraic Specification Language*. LNCS 2960. Springer, 2004.

[Mos05] T. Mossakowski. Heterogeneous specification and the heterogeneous tool set. University of Bremen, 2005. Habilitation thesis.

[MR95] U. Montanari and F. Rossi. Contextual nets. *Acta Informatica*, 32(6):545–596, 1995.

[MTHM97] R. Milner, M. Tofte, R. Harper, and D. MacQueen. *The Definition of Standard ML - Revised*. MIT Press, 1997.

[MTW87] B. Möller, A. Tarlecki, and M. Wirsing. Algebraic Specifications of Reachable Higher-Order Algebras. In D. Sannella and A. Tarlecki, editors, *Recent Trends in Data Type Specification, 5th Workshop on Abstract Data Types, Gullane, Scotland, September 1-4, 1987, Selected Papers*, LNCS 332, pages 154–169. Springer, 1987.

[Pad96] J. Padberg. *Abstract Petri Nets: A Uniform Approach and Rule-Based Refinement*. PhD thesis, Technical University Berlin, 1996. Shaker Verlag.

[PER95] J. Padberg, H. Ehrig, and L. Ribeiro. Algebraic High-Level Net Transformation Systems. *Mathematical Structures in Computer Science*, 5:217–256, 1995.

[Pet62] C.A. Petri. *Kommunikation mit Automaten*. PhD thesis, Schriften des Institutes für Instrumentelle Mathematik, Bonn, 1962.

[Poi86] A. Poigne. On specifications,theories, and models with higher types. *Inf. Control*, 68(1-3):1–46, 1986.

[PP01] F. Parisi-Presicce. On Modifying High Level Replacement Systems. *Electr. Notes Theor. Comput. Sci.*, 44(4), 2001.

[PU03] J. Padberg and M. Urbásek. Rule-Based Refinement of Petri Nets: A Survey. In H. Ehrig, W. Reisig, G. Rozenberg, and H. Weber, editors, *Petri Net Technology for Communication-Based Systems - Advances in Petri Nets*, LNCS 2472, pages 161–196. Springer, 2003.

[Rei85] Wolfgang Reisig. Petri nets with individual tokens. *Theoretical Computer Science*, 41:185–213, 1985.

[Rei87] H. Reichel. *Initial Computability, Algebraic Specifications and Partial Algebras*. Oxford Science Publications, 1987.

[Rei91] W. Reisig. Petri Nets and Algebraic Specifications. *Theoretical Computer Science*, 80:1–34, 1991.

[Roz97] G. Rozenberg, editor. *Handbook of graph grammars and computing by graph transformation: volume I. foundations*. World Scientific, 1997.

[RT86] G. Rozenberg and P.S. Thiagarajan. Petri Nets: Basic notions, structure, behaviour. In *Current Trends in Concurrency*, pages 585–668. LNCS 224, Springer, 1986.

[RWL+03] A. Ratzer, L. Wells, H. Lassen, M. Laursen, J. Qvortrup, M. Stissing, M. Westergaard, S. Christensen, and K. Jensen. CPN Tools for Editing, Simulating, and Analysing Coloured Petri Nets. In *24th Int. Conference on Applications and Theory of Petri Nets*, LNCS 2679, pages 450–462. Springer, 2003.

[San98] R. S. Sandhu. Role-Based Access Control. *Advances in Computers*, 46:237–286, 1998. Adademic Press.

[SB01] C. Silbertin-Blanc. The Hurried Philosophers. In G. Agha, F. De Cindio, and G. Rozenberg, editors, *Concurrent Object-Oriented Programming and Petri Nets*, pages 536–537. Springer, 2001.

[Sch89] H.W. Schmidt. *Specification and Correct Implementation of Nonsequential Systems Combining Abstract Data Types and Petri Nets.* PhD thesis, University Bremen, 1989. GMD-Bericht Nr. 176.

[Sch94] A.-W. Scheer. *Business Process Engineering, ARIS-Navigator for Reference Models for Industrial Enterprises.* Springer, 1994.

[SM02] L. Schröder and T. Mossakowski. HasCASL: towards integrated specification and development of functional programs. In Helene Kirchner and Christophe Ringeissen, editors, *Algebraic Methodology And Software Technology*, LNCS 2422, pages 99–116. Springer, 2002.

[Urb03] M. Urbášek. *Categorical Net Transformations for Petri Net Technology.* PhD thesis, Technische Universität Berlin, 2003.

[Val87] R. Valk. Nets in Computer Organisation. In W. Brauer, W. Reisig, and G. Rozenberg, editors, *Petri Nets: Central Models and Their Properties, Advances in Petri Nets, Part I*, LNCS 254, pages 377–396. Springer-Verlag, 1987.

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

[Val98] R. Valk. Petri Nets as Token Objects: An Introduction to Elementary Object Nets. In Jörg Desel and Manuel Silva, editors, *19th International Conference on Application and Theory of Petri Nets*, LNCS 1420, pages 1–25. Springer, 1998.

[Val01] R. Valk. Concurrency in Communicating Object Petri Nets. In G. Agha, F. de Cindio, and G. Rozenberg, editors, *Concurrent Object-Oriented Programming and Petri Nets*, LNCS 2001, pages 164–195. Springer, 2001.

[Vau86] J. Vautherin. Parallel Specification with Coloured Petri Nets and Algebraic Data Types. In *7th European Workshop on Application and Theory of Petri nets*, LNCS 266, pages 5–23. Springer, 1986.

[vdA98] W. M. P. van der Aalst. The Application of Petri Nets to Workflow Management. *Journal of Circuits, Systems, and Computers*, 8(1):21–66, 1998.

[vdAB02] W.M.P. van der Aalst and T. Basten. Inheritance of Workflows: An Approach to Tackling Problems Related to Change. *Theor. Comput. Sci.*, 270(1-2):125–203, 2002.

[Wie01] F. Wienberg. *Informations- und prozeorientierte Modellierung verteilter Systeme auf der Basis von Feature-Structure-Netzen*. PhD thesis, University Hamburg, 2001.

[Win87] G. Winskel. Petri nets, algebras, morphisms, and compositionality. *Information and Computation*, 72:197–238, 1987.

[Wol91] U. Wolter. An Algebraic Approach to Deduction in Equational Partial Horn Theories. *Elektronische Informationsverarbeitung und Kybernetik*, 27(2):85–128, 1991.

[Wol05] U. Wolter. Higher-Order Partial Algebras. Technical Report 299, University of Bergen (Norway), 2005.

# Appendix A

# Basic Notions of Category Theory

In this appendix, the notions and definition of category theory are explained so far as used in this thesis. Proofs can be found in [BW90] and [AHS90].

## A.1 Categorical Definitions and Constructions

**Definition A.1.1 (Category)**
A category $\mathbf{C} = (Ob_{\mathbf{C}}, Mor_{\mathbf{C}}, \circ, id)$ consists of

1. a class of objects $Ob_{\mathbf{C}}$,
2. for each pair of objects $A, B \in Ob_{\mathbf{C}}$ a set $Mor_{\mathbf{C}}(A, B)$ of morphisms, (a morphism $f \in Mor_{\mathbf{C}}(A, B)$ is also denoted by $f : A \longrightarrow B$),
3. for all objects $A, B, C \in Ob_{\mathbf{C}}$ a composition operator
$$\circ : Mor_{\mathbf{C}}(B, C) \times Mor_{\mathbf{C}}(A, B) \longrightarrow Mor_{\mathbf{C}}(A, C)$$
4. for each object $A \in Ob_{\mathbf{C}}$ there is a morphism $id_A \in Mor_{\mathbf{C}}(A, A)$

such that the following axioms are satisfied:

**Associativity** For all objects $A, B, C, D \in Ob_{\mathbf{C}}$ and for all morphisms $f \in Mor_{\mathbf{C}}(A, B)$, $g \in Mor_{\mathbf{C}}(B, C)$ and $h \in Mor_{\mathbf{C}}(C, D)$ we have
$$(h \circ g) \circ f = h \circ (g \circ f)$$
if at least one side is defined.

**Identity** For all objects $A, B \in Ob_{\mathbf{C}}$ and for all morphisms $f \in Mor_{\mathbf{C}}(A, B)$ we have
$$f \circ id_A = f \text{ and } id_B \circ f = f.$$

**Definition A.1.2 (Isomorphism)**
Let $\mathbf{C}$ be a category. A morphism $f \in Mor_{\mathbf{C}}(A, B)$ is an isomorphism if there exists a morphism $g \in Mor_{\mathbf{C}}(B, A)$ such that $f \circ g = id_B$ and $g \circ f = id_A$. In this case, the objects $A$ and $B$ are said to be isomorphic, written $A \cong B$.

**Definition A.1.3 (Initial Object)**
Let $\mathbf{C}$ be a category. An object $I \in Ob_{\mathbf{C}}$ is an initial object in $\mathbf{C}$ if for any object $A \in Ob_{\mathbf{C}}$ there exists a unique morphism $f \in Mor_{\mathbf{C}}(I, A)$.

**Definition A.1.4 (Coproducts)**
A coproduct of two objects $A$ and $B$ in a category $\mathbf{C}$ is an object $A + B$ together with two injection morphisms $in_A \in Mor_{\mathbf{C}}(A, A + B)$ and $in_B \in Mor_{\mathbf{C}}(B, A + B)$ such that for any object $D$ in $\mathbf{C}$ and pair of morphisms $f \in Mor_{\mathbf{C}}(A, D)$ and $g \in Mor_{\mathbf{C}}(B, D)$, there exists one and only one morphism $k \in Mor_{\mathbf{C}}(A + B, D)$ such that $k \circ in_A = f$ and $k \circ in_B = g$.

**Definition A.1.5 (Coequalizer)**
A coequalizer of a pair of morphisms $f \in Mor_{\mathbf{C}}(A, B)$ and $g \in Mor_{\mathbf{C}}(A, B)$ is a morphism $h \in Mor_{\mathbf{C}}(C, B)$ such that $f \circ h = g \circ h$ and for any object $X$ in $\mathbf{C}$ and morphism $i \in Mor_{\mathbf{C}}(X, A)$ with $f \circ i = g \circ i$, there exists one and only one morphism $k \in Mor_{\mathbf{C}}(X, C)$ such that $i = h \circ k$.

**Definition A.1.6 (Pushouts)**
A pushout of a pair of morphisms $f \in Mor_{\mathbf{C}}(A, B)$ and $g \in Mor_{\mathbf{C}}(A, C)$ is an object $D$ in $\mathbf{C}$ and a pair of morphisms $f' \in Mor_{\mathbf{C}}(B, D)$ and $g' \in Mor_{\mathbf{C}}(C, D)$ such that $f' \circ f = g' \circ g$ and for any object $X$ in $\mathbf{C}$ and pair of morphisms $i \in Mor_{\mathbf{C}}(B, X)$ and $j \in Mor_{\mathbf{C}}(C, X)$ with $i \circ f = j \circ g$, there exists one and only one morphism $k \in Mor_{\mathbf{C}}(D, X)$ such that $i = f' \circ k$ and $j = g' \circ k$.

**Definition A.1.7 (Pullbacks)**
A pullback of a pair of morphisms $f \in Mor_{\mathbf{C}}(B, D)$ and $g \in Mor_{\mathbf{C}}(C, D)$ is a object $A$ in $\mathbf{C}$ and a pair of morphisms $f' \in Mor_{\mathbf{C}}(A, B)$ and $g' \in Mor_{\mathbf{C}}(A, C)$ such that $f' \circ f = g' \circ g$ and for any object $X$ in $\mathbf{C}$ and pair of morphisms $i \in Mor_{\mathbf{C}}(X, B)$ and $j \in Mor_{\mathbf{C}}(X, C)$ with $f \circ i = g \circ j$, there exists one and only one morphism $k \in Mor_{\mathbf{C}}(X, D)$ such that $i = k \circ f'$ and $j = k \circ g'$.

**Definition A.1.8 (Finite Cocompleteness)**
A category $\mathbf{C}$ with the following properties

1. $\mathbf{C}$ has an initial object,
2. every pair of objects has a coproduct, and
3. every parallel pair of morphisms has an coequalizer

has all finite colimits.

**Fact A.1.9 (Finite Cocompleteness)**
A category $\mathbf{C}$ that has an initial object and all pushouts is finitely cocomplete. □

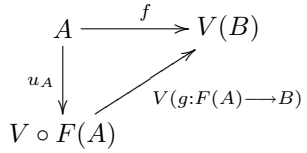
## A.2 Constructions Based on Functors

**Definition A.2.1 (Functor)**
Let $\mathbf{C}$ and $\mathbf{D}$ be categories. A functor $F : \mathbf{C} \longrightarrow \mathbf{D}$ is a pair of mappings $F_{Ob} : Ob_{\mathbf{C}} \longrightarrow Ob_{\mathbf{D}}$ and $F_{Mor} : Mor_{\mathbf{C}} \longrightarrow Mor_{\mathbf{D}}$ for which

1. if $f \in Mor_{\mathbf{C}}(A, B)$, then $F_{Mor}(f) \in Mor_{\mathbf{D}}(F_{Ob}(A), F_{Ob}(B))$,
2. $F_{Mor}(g \circ f) = F_{Mor}(g) \circ F_{Mor}(f)$ whenever $f \circ g$ is defined, and
3. $F_{Mor}(id_A) = id_{F_{Ob}(A)}$ for each object $A \in Ob_{\mathbf{C}}$.

If no confusion arises, we will denote both $F_{Ob}$ and $F_{Mor}$ by $F$.

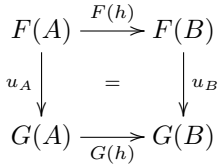
**Definition A.2.2 (Free Construction)**
Let $\mathbf{C}$ and $\mathbf{D}$ be categories, $V : \mathbf{D} \longrightarrow \mathbf{C}$ a functor and $A \in Ob_{\mathbf{C}}$ an object. We call and object $F(A) \in Ob_{\mathbf{C}}$ free construction of $A$ wrt. $V$ if there is a morphism $u_A : A \longrightarrow V \circ F(A)$, called universal morphism, which satisfies the following property, called universal property. For each object $B \in \mathbf{D}$ and each morphism $f : A \longrightarrow V(B)$ in $\mathbf{C}$ there is a unique morphism $g : F(A) \longrightarrow B$ in $\mathbf{D}$ such that $V(g) \circ u_A = f$. In this case we say that the following diagram commutes:



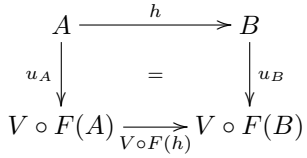

**Definition A.2.3 (Natural Transformation)**
Let Let $\mathbf{C}$ and $\mathbf{D}$ be categories and $F, G : \mathbf{C} \longrightarrow \mathbf{D}$ be functors. A family

$$u = (u_A : F(A) \longrightarrow G(A))_{A \in Ob_{\mathbf{C}}}$$

of morphisms in $\mathbf{C}$ is called a natural transformation, written $u : F \longrightarrow G$ if for all morphisms $h : A \longrightarrow B$ in $\mathbf{C}$ the following diagram commutes:




**Fact A.2.4 (Free Functor)**
Given a free construction $F(A)$ over $A$ wrt. $V$ for $V : \mathbf{D} \longrightarrow \mathbf{C}$ and all $A \in Ob_{\mathbf{C}}$, then this free construction can be extended to morphisms in $\mathbf{C}$ such that for each $h : A \longrightarrow B$ in $\mathbf{C}$ $F(h)$ is uniquely defined by commutativity of the following diagram:




In this way we obtain a functor $F : \mathbf{C} \longrightarrow \mathbf{D}$, called free functor wrt. $V$, and a natural transformation $u : ID_{\mathbf{C}} \longrightarrow V \circ F$, called universal transformation, where $ID_{\mathbf{C}}$ is the identical functor. The free functor $F : \mathbf{C} \longrightarrow \mathbf{D}$ wrt. $V$ is usually called left adjoint to $V$ and dually $V$ is in this case right adjoint to $F$, written $F \vdash V$. □

**Theorem A.2.5 (Preservation of Colimits)**
Let $F : \mathbf{C} \longrightarrow \mathbf{D}$ and $V : \mathbf{D} \longrightarrow \mathbf{C}$ be functors such that $F \vdash V$. Then $F$ preserves colimits. □

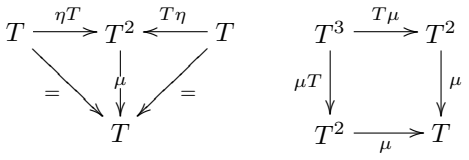
**Definition A.2.6 (Monad)**
A monad on a category $\mathbf{C}$ is a triple $\mathbf{T} = (T, \eta, \mu)$ consisting of a functor

$$T : \mathbf{C} \longrightarrow \mathbf{C}$$

and natural transformations

$$\eta : ID_{\mathbf{C}} \longrightarrow T \text{ and } \mu : T^2 \longrightarrow T$$

such that the following diagrams commute:




**Fact A.2.7 (Kleisli Category)**
Let $\mathbf{T} = (T, \eta, \mu)$ be a monad on the category $\mathbf{C}$. Then the Kleisli category $\mathbf{K(T)}$ has the same objects as $\mathbf{C}$. For $A, B \in Ob_{\mathbf{C}}$ we have $Mor_{\mathbf{K(T)}}(A, B) = Mor_{\mathbf{C}}(A, T(B))$. The identity of an object $A$ is the arrow $\eta_A : A \longrightarrow T(A)$. The composition of morphisms is as follows. If $f : A \longrightarrow T(B)$ and $g : B \longrightarrow T(C)$ are arrows in $\mathbf{C}$ , we let their composite in $\mathbf{K(T)}$ be the following composite in $\mathbf{C}$ :

$$A \xrightarrow{f} T(B) \xrightarrow{T(g)} T^2(C) \xrightarrow{\mu_C} T(C).$$

Moreover, there are functors $V : \mathbf{K(T)} \longrightarrow \mathbf{C}$ and $F : \mathbf{C} \longrightarrow \mathbf{K(T)}$ defined by $V(A) = T(A)$ and $V(f) = \mu_A \circ T(f)$, where $A$ is the domain of $f$, and $F(A) = A$ and for $g : A \longrightarrow B$, $F(g) = T(g) \circ \eta_A$. Then $F$ is left adjoint to $U$ and $T = U \circ F$. □

# Appendix B

# Free Commutative Monoids

In this section we will summarize necessary notions about free commutative monoids because they are frequently used in this thesis.

## B.1 Definitions of Free Commutative Monoids

**Definition B.1.1 (Category CMon)**
A commutative monoid $(M, \bullet, \epsilon)$ consists of

- s set $M$ which is closed wrt. $\bullet$,
- a neutral element $\epsilon$,
- an operation $\bullet : M \times M \longrightarrow M$ such that for all $w_1, w_2, w_3 \in M$ we have
  1. $\epsilon \bullet w_1 = w_1$ and $w_1 \bullet \epsilon = w_1$,
  2. $w_1 \bullet w_2 = w_2 \bullet w_1$, and
  3. $(w_1 \bullet w_2) \bullet w_3 = w_1(\bullet w_2 \bullet w_3)$.

A function $h : M_1 \longrightarrow M_2$ is a monoid homomorphism

$$h : (M_1, \bullet_1, \epsilon_1) \longrightarrow (M_2, \bullet_2, \epsilon_2)$$

if for all $w_1, w_2 \in M$

1. $h(\epsilon_1) = \epsilon_2$ and
2. $h(w_1 \bullet_1 w_2) = h(w_1) \bullet_2 h(w_2)$.

The category **CMon** consists of commutative monoids as objects and monoid homomorphisms as morphisms.

**Definition B.1.2 (Free Commutative Monoid)**
Let $P$ be a set, then $(P^{\oplus}, \lambda, \oplus)$ is called the free commutative monoid generated by $P$, s.t. for all $u, v, w \in P^{\oplus}$ the following equations hold:

- $\lambda \in P^{\oplus}$
- $P \subseteq P^{\oplus}$
- $u, v \in P^{\oplus} \Longrightarrow u \oplus w \in P^{\oplus}$
- $v \oplus \lambda = \lambda \oplus v = v$

- $u \oplus (v \oplus w) = (u \oplus v) \oplus w$
- $v \oplus w = w \oplus v$

Elements of the free commutative monoid $w \in P^{\oplus}$ for some set $P$ can be represented as $w = \sum_{i=1}^{n} k_i \cdot p_i$ with coefficient $k_i \in \mathbb{N}$. For $n = 0$ we have $\sum_{i=1}^{0} k_i \cdot p_i = \lambda$.

In the computer science literature such elements are called finite multisets, i.e. multisets $w : P \longrightarrow \mathbb{N}$ for which the set $\{p|w(p) \neq 0\}$ is finite. A multiset $w$ is usually represented by the formal sum $\sum_{p \in P} w(p) \cdot p$. We denote by $\mathcal{M}(P)$ the set of all finite multisets over $P$.

An element $w = \sum_{i=1}^{n} k_i \cdot p_i \in P^{\oplus}$ with $k_i \in \mathbb{N}, p_i \in P$ is in normal form if $k_i > 0$ for all $i = 1, \ldots, n$ and $p_i \neq p_j$ for $i \neq j$. Then $P^{\oplus}$ is defined by

$$P^{\oplus} = \{w | w = \sum_{i=1}^{n} k_i \cdot p_i, w \text{ in normal form}\}$$

and we have $P^{\oplus} \cong \mathcal{M}(P)$. So we can switch between the two different notation of formal sums given above.

Let $f : P_1 \longrightarrow P_2$ be a function. The extension $f^{\oplus} : P_1^{\oplus} \longrightarrow P_2^{\oplus}$ is the set-based monoid homomorphism defined for all $w \in P^{\oplus}$ by

$$\begin{aligned} f^{\oplus}(w) &= f^{\oplus}(\textstyle\sum_{p_1 \in P_1} w(p_1) \cdot p_1) \\ &= \textstyle\sum_{p_1 \in P_1} w(p_1) \cdot f(p_1) \\ &= \textstyle\sum_{p_2 \in P_2} (\sum_{\substack{p_1 \in P_1 \\ f(p_1) = p_2}} w(p_1)) \cdot p_2 \end{aligned}$$

**Definition B.1.3 (Forgetful Functor $V_{\mathbf{CMon}}$)**
The forgetful functor $V_{\mathbf{CMon}} : \mathbf{CMon} \longrightarrow \mathbf{Sets}$ sends a monoid $(M, \bullet, \epsilon)$ to the set $M$ and each monoid homomorphism $h : (M_1, \bullet_1, \epsilon_1) \longrightarrow (M_2, \bullet_2, \epsilon_2)$ to the corresponding function $h : M_1 \longrightarrow M_2$.

**Definition B.1.4 (Free Functor $F_{\mathbf{CMon}}$)**
Let $P$ be a set. Then for each function $f : P \longrightarrow V_{\mathbf{CMon}}(M, \bullet, \epsilon)$ we get a unique monoid homomorphism $f^{\oplus} : (P^{\oplus}, \lambda, \oplus) \longrightarrow (M, \bullet, \epsilon)$ such that $f = V_{\mathbf{CMon}}(f^{\oplus}) \circ u$ with inclusion $u : P \longrightarrow P^{\oplus}$. The free construction can be extended to a free functor $F_{\mathbf{CMon}} : \mathbf{Sets} \longrightarrow \mathbf{CMon}$.

**Remark B.1.5 (Normal Form)**
We use the notation $\sum_{i=1}^{n} p_i$ with $p_i \in P$ as an abbreviation for $p_1 \oplus \ldots \oplus p_n, n \in \mathbb{N}$, where the elements $p_i$ of $P$ are not necessary disjoint, i.e. $\sum_{i=1}^{n} p_i$ might not be in normal form. To transform this sum into normal form we first observe that each element $p_i$ itselfs represents a multiset $w_i : P \longrightarrow \mathbb{N}$ for all $i \in \{1, \ldots, n\}$ with $w_i(p_i) = 1$ and $w_i(p) = 0$ for all $p \in P$ and $p \neq p_i$. Then we have

$$\sum_{i=1}^{n} p_i = \sum_{p \in P} \sum_{\substack{i \,=\, 1 \\ p_i \,=\, p}}^{n} w_i(p_i) \cdot p.$$

**Definition B.1.6 (Induced Function by Free Commutative Monoid)**
Free commutative monoids imply the operations $\oplus, \ominus, \leq, \geq, =, \neq$ and $\in$ on linear sums. These are the obvious addition, subtraction and comparison of coefficients on

linear sums. Let $w_1, w_2 \in P^{\oplus}$ with $w_1 = \sum_{p \in P} w_1(p) \cdot p$ and $w_2 = \sum_{p \in P} w_2(p) \cdot p$, then

**(1)** $p \in w_1$ if $w_1(p) \geq 0$

**(2)** $w_1 \leq w_2$ if $\forall p \in P : w_1(p) \leq w_2(p)$

**(3)** $w_1 \geq w_2$ if $\forall p \in P : w_1(p) \geq w_2(p)$

**(4)** $w_1 = w_2$ if $\forall p \in P : w_1(p) = w_2(p)$

**(5)** $w_1 \neq w_2$ if $\exists p \in P : w_1(p) \neq w_2(p)$

**(6)** $w_1 \oplus w_2(p) = (w_1(p) + w_2(p))$ for all $p \in P$

**(7)** $w_1 \ominus w_2(p) = (w_1(p) - w_2(p)$ for all $p \in P$ if $w_1 \geq w_2$

## B.2 Restrictions and Cartesian Products

**Definition B.2.1 (Restrictions on Free Commutative Monoids)**
Given $w : P \longrightarrow \mathbb{N}$ and let $i : P_1 \longrightarrow P$ be an inclusion. Then we define

$$w_{|P_1} = w \circ i.$$

Moreover there is a special case given by

$$P_1 = \{p\} \text{ denoted with } w_{|p}.$$

**Fact B.2.2 (Restrictions are Compatible with Monoid-Operations)**
Let $P_1 \subseteq P$ and $w, w' \in P^{\oplus}$. Then we have

**(1)** $(w \oplus w')_{|P_1} = w_{|P_1} \oplus w'_{|P_1}$

**(2)** $(w \ominus w')_{|P_1} = w_{|P_1} \ominus w'_{|P_1}$

**(3)** $w \leq w' \Longrightarrow w_{|P_1} \leq w'_{|P_1}$.

**Proof:** Let $w, w' : P \longrightarrow \mathbb{N}$ and let $i : P_1 \longrightarrow P$ be an inclusion.

**Proof of (1):** For all $p \in P_1$ we have

$$\begin{array}{rcl} (w \oplus w')_{|P_1}(p) & = & (w \oplus w')(i(p)) \\ & = & (w \circ i(p)) + (w' \circ i(p)) \\ & = & (w_{|P_1}(p)) + (w'_{|P_1}(p)) \\ & = & (w_{|P_1} \oplus w'_{|P_1})(p). \end{array}$$

**Proof of (2):** analogously

**Proof of (3):**

$$\begin{array}{rl} & w(p) \leq w'(p) \text{ for all } p \in P \\ \Longrightarrow & w(p) \leq w'(p) \text{ for all } p \in P_1 \\ \Longleftrightarrow & w \circ i(p) \leq w' \circ i(p) \text{ for all } p \in P_1 \\ \Longleftrightarrow & w_{|P_1} \leq w'_{|P_1}. \end{array}$$

□

**Definition B.2.3 (Cartesian Products of Free Commutative Monoid)**
Free commutative monoids imply the operations of Cartesian products $\times$ on linear sums. Let $w \in A^{\oplus}, w' \in B^{\oplus}$ then we define for all $(a, b) \in A \times B$

$$w \times w'(a, b) = w(a) \cdot w'(b).$$

Let $w \in (A \times B)^{\oplus}$ for some sets $A$ and $B$. Then for $A_1 \subseteq A$ and $B_1 \subseteq B$ there is an inclusion $i : (A_1 \times B_1) \longrightarrow (A \times B)$ and we have due to Def. B.2.1

$$w_{|(A_1 \times B_1)} = w \circ i.$$

For $i : (A \times B_1) \longrightarrow (A \times B)$ we denote $w_{|(A \times B_1)}$ by $w_{|B_1}$. If no confusion arises, we also denote for $i : (A \times \{b\}) \longrightarrow (A \times B)$ the restriction $w_{|(A \times \{b\})}$ by

$$w_{|b} = \sum_{a \in A} w(a, b) \cdot a.$$

**Fact B.2.4 (Restrictions are Compatible with Cartesian Products)**
Let $w \in A^{\oplus}, w' \in B^{\oplus}$, $A_1 \subseteq A$, and $B_1 \subseteq B$. Then we have

$$(w \times w')_{|(A_1 \times B_1)} = w_{|A_1} \times w'_{|B_1}$$

**Proof:** Let $i : (A_1 \times B_1) \longrightarrow (A \times B)$ be an inclusion. Then there are inclusions $i_A : A_1 \longrightarrow A$ and $i_B : B_1 \longrightarrow B$ so that we have for all $(a, b) \in A_1 \times B_1$:

$$\begin{array}{rl} & (w \times w')_{|(A_1 \times B_1)}(a, b) \\ = & (w \times w')(i(a, b)) \\ = & (w \times w')(i_A(a), i_B(b)) \\ = & (w \circ i_A(a)) \cdot (w' \circ i_B(b)) \\ = & (w_{|A_1}(a)) \cdot (w'_{|B_1}(b)) \\ = & (w_{|A_1} \times w'_{|B_1})(a, b). \end{array}$$

□

# Appendix C

# Elementary Object Systems

In the following we review the well-known notion of elementary net systems, i.e. elementary nets with an initial marking.

## C.1 Definitions of Elementary Net Systems

**Definition C.1.1 (Elementary Nets)**
An elementary net $N = (B, E, F)$ is defined by a finite set of places (or conditions) $B$, a finite set of transitions (or events) $E$, disjoint from $B$, and a flow relation $F \subseteq (B \times E) \cup (E \times B)$.

**Definition C.1.2 (Firing Behavior of EN-Systems)**
Given an elementary net $N = (B, E, F)$, then

1. a marking of $N$ is given by $m \subseteq B$,
2. a transition $e \in B$ is $C$-enabled for a marking $m \subseteq B$, denoted by $m[e\rangle$ or $m \xrightarrow{t}$ if we have $\bullet e \subseteq m$ and $e\bullet \subseteq B \setminus m$ and
3. if $e \in B$ is $m$-enabled the follower marking $m'$ is given by
$$m' = (m \setminus \bullet e) \cup e\bullet$$
and denoted by $m[e\rangle m'$ or $m \xrightarrow{t} m'$.

These notions are usually extended to words $w \in E^+$ and written $m \xrightarrow{e} m'$.

**Definition C.1.3 (Elementary Net Systems)**
An elementary net (EN) system $N = (B, E, F, m^0)$ is defined by an elementary net $N = (B, E, F)$ and an initial marking $m^0 \subseteq B$. Moreover, we have

- the set of firing or occurrence sequences of $N$ defined by
$$FS(N) := \{w \in E^* | m^0 \xrightarrow{w}\}$$
- the set of reachable markings of $N$, also called reachable set of $N$, defined by
$$R(N) := \{m \subseteq B | \exists w \in E^* : m^0 \xrightarrow{w} m\}.$$

**Definition C.1.4 ((Structural) State Machine)**
Given an elementary net $N = (B, E, F)$, then it is called a structural state machine if each transition $e \in E$ has exactly one input place ($|\bullet e| = 1$) and exactly one output place ($|e \bullet| = 1$). $N$ is said to be a state machine if it is a structural state machine and $m^0$ contains exactly one token ($|m^0| = 1$).

Using the notation of "Petri nets are Monoids" [MM90] the definition above are equivalent to the following definition.

**Definition C.1.5 (Elementary Nets)**
An elementary net $N = (B, E, pre, post)$ consists of

- finite set of places (or conditions) $B$, a finite set of transitions (or events) $E$, disjoint from $B$,
- pre- and post conditions $pre, post : E \longrightarrow \mathcal{P}(B)$ assigning to each transition $e \in E$ an element of the powerset $\mathcal{P}(B)$.

**Definition C.1.6 (Firing Behavior of EN-Systems)**
Given an elementary net $N = (B, E, pre, post)$, then

1. a marking of $N$ is given by $m \subseteq B$,
2. a transition $e \in B$ is $m$-enabled for a marking $m \in\subseteq B$, denoted by $m[e\rangle$ if we have $pre(t) \subseteq m$ and $post(t) \subseteq B \setminus m$ and
3. if $e \in B$ is $m$-enabled the follower marking $m'$ is given by

$$m' = (m \setminus pre(t)) \cup post(t)$$

and denoted by $m[e\rangle m'$.

**Definition C.1.7 (Elementary Net Systems)**
An elementary net (EN) system $N = (B, E, pre, post, m^0)$ consists of an elementary net $(B, E, pre, post)$ and an initial marking $m^0 \subseteq \mathcal{P}(B)$.

## C.2 Definitions of Elementary Object Systems

In the following we review the notions of elementary objects systems as defined in [Val98, Val01]. Elementary object systems are an extension of unary elementary object systems in such a way that different object nets move through in a system net and interact with both, the system net and with other object nets.

**Definition C.2.1 (Elementary Object System)**
A elementary object system is a tuple $EOS = (SN, \hat{ON}, Rho, type, \hat{\mathbf{M}})$ where

- $SN = (P, T, W)$ is net (i.e. an EN system without initial marking) called system net of $EOS$,
- $\hat{ON} = \{ON_1, \ldots, ON_n\}$ $(n \geq 1)$ is a finite set of EN systems called object nets of $EOS$, denoted by $ON_i = (B_i, E_i, F_i, \mathbf{m_{0i}})$ is an EN system called object net of $EOS$ and
- $Rho = (\rho, \sigma)$ is the interaction relation, consisting of a system/object interaction relation $\rho \subseteq T \times \mathbf{E}$ where $\mathbf{E} := \bigcup\{E_i | 1 \leq i \leq n\}$ and a symmetric object/object interaction relation $\sigma \subseteq (\mathbf{E} \times \mathbf{E}) \setminus id_E$,
- $type : W \longrightarrow 2^{\{1,\ldots,n\}} \cup \mathbb{N}$ is the arc type function and
- $\hat{\mathbf{M}}$ is a marking as defined in Def. C.2.2.

**Definition C.2.2 (Marking of Elementary Object Systems)**
The set $\mathbf{Obj} := \{(ON_i, \mathbf{m_i}) | 1 \leq i \leq n, \mathbf{m_i} \in R(ON_i)\}$ is the set of objects of the $EOS$. An object-marking (O-marking) is a mapping $\hat{\mathbf{M}} : P \longrightarrow 2^{\mathbf{Obj}} \cup \mathbb{N}$ such that $\hat{\mathbf{M}}(p) \cap \mathbf{Obj} \neq \emptyset \Longrightarrow \hat{\mathbf{M}}(p) \cap \mathbb{N} = \emptyset$ for all $p \in P$.

**Definition C.2.3 (Components of Elementary Object Systems)**
Let $EOS = (SN, \hat{ON}, Rho, type, \hat{\mathbf{M}})$ be an elementary object system, but in some arbitrary marking $\hat{\mathbf{M}}$.

- $Rho = (\rho, \sigma)$ is said to be separated if $i \sigma j \Longrightarrow \rho i = \emptyset = \rho j$.
- The $i$-component $(1 \leq i \leq n)$ of $EOS$ is the $EN$ system

  $$SN(i) = (P, T, W(i), \mathbf{M}_{0i}(p))$$

  defined by $W(i) = \{(x, y) | i \in type(x, y)\}$ and $\mathbf{M}_{0i}(p) = 1$ iff $(ON_i, \mathbf{m_i}) \in \hat{\mathbf{M}}(p)$. The 0-component (zero-component) is the P/T-net

  $$SN(0) = (P, T, W(0), \mathbf{M}_{00}(p))$$

  with the arc weight function $W(0)(x, y) = k$ if $type(x, y) = k \in \mathbb{N}$ and $\mathbf{M}_{00}(p) = k \in \mathbb{N}$ iff $k \in \hat{\mathbf{M}}(p)$.
- The subnet $SN(1 \ldots n) = (P, T, W(1 \ldots n), \mathbf{M}_{1 \ldots n}(p))$ where $W(i \ldots n) = \bigcup \{W(i) | 1 \leq i \leq n\}$ and $\mathbf{M}_{1 \ldots n}(p) = \hat{\mathbf{M}}(p) \cap \mathbf{Obj}$ is said to be the object-component.
- $EOS$ is said to be a simple elementary object system if $SN(i \ldots n)$ is a structural state machine, all $i$-components of $SN$ are state machines and $Rho$ is separated.

There are some inconsistencies in the definitions of elementary object systems and their markings. On the one hand, the system net $SN$ of an elementary object system $EOS$ is rather a P/T-net than an EN-net because the zero-component $SN(O)$ is a P/T-net (see Def. C.2.2). On the other hand, there is an ambiguity concerning the marking of an object $(ON_i, m_i) \in \mathbf{Obj}$ of an elementary object system $EOS$ (see Def. C.2.2) because there is an initial marking $m_{0i}$ and a reachable marking $m_i$ in the object $(ON_i = (B_i, E_i, F_I, m_{0i}), m_i)$ (although it is clear due to the explanations in [Val98, Val01] that the actual marking of an object is meant to be the reachable marking $m_i$).

It is not explicitly mentioned in the following definition but implicitly assumed that the object component $SN(1 \ldots n)$ of an elementary object system $EOS$ is a structural state machine. Otherwise dropping this condition would lead to inconsistencies in the definition of the dynamical behavior of elementary object systems (see [Val98, Val01]). Thus the elementary object systems in the following definition do not reflect fork/join-control structures.

**Definition C.2.4 (Firing Behavior of Elementary Object Systems)**
Let $EOS = (SN, \hat{ON}, Rho, type, \hat{\mathbf{M}})$ be an elementary object system and $\hat{\mathbf{M}} : P \longrightarrow 2^{\mathbf{Obj}} \cup \mathbb{N}$ be an O-marking and $t \in T, e_i \in E_i, e_j \in E_j, i \neq j$.

1. Transition $t \in T$ is activated in $\hat{\mathbf{M}}$ (denoted $\hat{\mathbf{M}} \xrightarrow{t}$) if $t\rho = \emptyset$ and the following holds:
   (a) $t$ is activated in the zero-component of $SN$ (i.e. in the P/T-net part)

(b) By the state machine property there is at most one type $i \in \{1, \ldots, n\}$ such that $i \in type(p_1, t)$ and $i \in type(t, p_2)$ for some $p_1 \in \bullet t$ and $p_2 \in t\bullet$. In this case there must be some object $(ON_i, \mathbf{m_i}) \in \hat{\mathbf{M}}(p_1)$.

If $t$ is activated, then $t$ may occur ($\hat{\mathbf{M}} \xrightarrow{t} \hat{\mathbf{M}}'$) and the follower marking $\hat{\mathbf{M}}'$ is defined as follows: with respect to the zero-components tokens are changed according to the ordinary P/T-net occurrence rule. In case 1.2. $(ON_i, \mathbf{m_i})$ is removed from $p_1$ and added to $p_2$ (only if $p_1 \neq p_2$).

2. A pair $(t, e) \in T \times E_i$ with $t \rho e$ is activated in $\hat{\mathbf{M}}$ (denoted $\hat{\mathbf{M}} \xrightarrow{(t,e)}$) if in addition to case 1 transition $e$ is also activated for $ON_i$ in $\mathbf{m_i}$. Instead of $(ON_i, \mathbf{m_i})$ the changed object $(ON_i, \mathbf{m_{i+1}})$ where $\mathbf{m_i} \xrightarrow{e} \mathbf{m_{i+1}}$ is added.

3. A pair $(e_i, e_j) \in E_i \times E_j$ with $e_i \sigma e_j$ is activated in $\hat{\mathbf{M}}$ (denoted $\hat{\mathbf{M}} \xrightarrow{(e_i, e_j)}$) if for some place $p \in P$ two objects $(ON_i, \mathbf{m_i}) \in \hat{\mathbf{M}}(p)$ and $(ON_j, \mathbf{m_j}) \in \hat{\mathbf{M}}(p)$ are in the same place $p$ and $\mathbf{m_i} \xrightarrow{e_i} \mathbf{m_{i+1}}$ and $\mathbf{m_j} \xrightarrow{e_j} \mathbf{m_{j+1}}$. In the follower marking $\hat{\mathbf{M}}'$ the objects $(ON_i, \mathbf{m_i})$ and $(ON_j, \mathbf{m_j})$ in $p$ are replaced by $(ON_i, \mathbf{m_{i+1}})$ and $(ON_j, \mathbf{m_{j+1}}$, resp.

4. A transition $e \in E_i$ with $e\sigma = \sigma e = \emptyset$ is activated in $\hat{\mathbf{M}}$ (denoted $\hat{\mathbf{M}} \xrightarrow{e}$) if for some place $p \in P$ we have $(ON_i, \mathbf{m_i}) \in \hat{\mathbf{M}}(p)$ and $\mathbf{m_i} \xrightarrow{e_i} \mathbf{m_{i+1}}$. In the follower marking $\hat{\mathbf{M}}'$ the object $(ON_i, \mathbf{m_i})$ is replaced by $(ON_i, \mathbf{m_{i+1}})$.

# Appendix D

# Notation

**Basics**

| | |
|---|---|
| $\mathbb{N}$ | natural numbers |
| $\mathcal{P}_{fin}$ | (finite) power sets |

**Mathematical preliminaries**

| | |
|---|---|
| $\longrightarrow$ | total function |
| $\multimap$ | partial function |
| $\otimes$ | Cartesian product respecting sorts/types |

**Free commutative monoids**

| | |
|---|---|
| $P^{\oplus}$ | free commutative monoid over the set $P$ |
| $\oplus, \ominus$ | operations on monoid elements |
| $\leq, \geq$ | comparison predicates on monoid elements |
| $w_{\mid P}$ | restrictions of monoid elements |

**Algebraic higher-order nets**

| | |
|---|---|
| $M[(t,v)\rangle$ | transition $t$ is enabled in marking $M$ under variable valuation $v$ |
| $M[(t,v)\rangle M'$ | firing of transition $t$ from marking $M$ to follower marking $M'$ |
| $[M\rangle$ | set of reachable markings of $M$ |

**Categories**

| | |
|---|---|
| **Sets** | sets and functions |
| **CMon** | commutative monoids and monoid homomorphisms |
| **Sig** | classical signatures and signature morphisms |
| **Alg**($\Sigma$) | classical $\Sigma$-algebras and homomorphisms |
| **Alg**($SP$) | classical $SP$-algebras and homomorphisms |
| **HOSig** | higher-order signatures and signature morphisms |
| **HOAlg**($\Sigma$) | higher-order partial $\Sigma$-algebras and homomorphisms |
| **PTNet** | P/T-nets and P/T-net morphisms |
| **PTSys** | P/T-systems and P/T-system morphisms |
| **AHLNet** | AHL-nets and AHL-net morphisms |
| **AHOS** | AHO-net schemes and AHO-net scheme morphisms |
| **AHON** | AHO-nets and AHO-net morphisms |