Reverse Petri Net Technology Transfer: On the Boundary of Theory and Applications[a]

Hartmut Ehrig[b] Maike Gajewsky[b] Sabine Lembke[b] Julia Padberg[b]
Volker Gruhn[c]

Topics: High-level net model, structuring techniques, application to workflow management, technology transfer

Abstract

The housing management system project WIS of the company LION is one of the most successful practical applications of Petri net technology in the area of cooperating and distributed business process management. This system is based on the workflow management environment LEU and the FunSoft approach. FunSoft nets are very powerful in practice, but they are not well understood from a theoretical point of view. In this paper we study the essential parts of FunSoft and present a comprehensive parameterized abstraction, called ParFun. ParFun is generic concerning the data and the organizational model. In addition to specific FunSoft features our approach comprises general concepts of union, fusion, and rule-based refinement in a clean algebraic framework including interesting compatibility results. In this sense ParFun is a new high-level net approach motivated by FunSoft and the practical application WIS. It allows new theoretical concepts and results and - via different instantiations of the parameters - also new practical applications. Hence, ParFun is the result of reverse Petri net technology transfer on the boundary of theory and application.

1 Introduction

Petri nets have been studied for more than three decades in the scientific community with deep theoretical results and have also been successful in numerous practical applications, whereas other formal specification techniques with profound semantical basis, like algebraic specification [EM85], are less used in practice up to now. The attractiveness of Petri nets is due to the natural graphical representation of nets and the intuitive token game which allows easy comprehension of concurrent processes in all kinds of applications (see [DO96]). One of these successful application areas of Petri net technology is the development of distributed information systems [OSS94] and cooperating and distributed business process management [GGK96]. A typical important application is the housing management system WIS of the company LION based on FunSoft nets, which has been developed in 20 project groups for 8 different clients in 160 person years [GGK96].

[a]This work is part of the common research project "DFG-Forschergruppe Petrinetz-Technologie" between H. Weber (Coordinator), H. Ehrig (both from the Technical University Berlin) and W. Reisig (Humboldt University Berlin), supported by the Deutsche Forschungsgemeinschaft (DFG).

[b]Technical University Berlin, e-mail: {ehrig,magda,lembkes,padberg}@cs.tu-berlin.de
[c]University Dortmund, e-mail: gruhn@ls10.informatik.uni-dortmund.de

From the theoretical computer science point of view Petri nets are well understood, at least for the classical low-level nets (see e.g. [Rei85, NRT92, DE95]). For practical applications even more important are high-level nets, especially predicate/transition nets [GL81], coloured Petri nets [Jen81], [Jen92] and algebraic nets [Rei91] and several variants of these net classes. Although there are already several interesting results for high-level nets there is still a gap between theory and praxis. On the one hand there are several theoretical concepts and results which are rarely used in practice, on the other hand several variants of high-level nets are used in practice which are not well understood from a theoretical point of view. A typical example of this kind are FunSoft nets developed by V. Gruhn [Gru91]. We have pointed out already that the commercial business process management environment LEU [GW95] based on FunSoft has been very successful in the development of the housing management system WIS. In order to make FunSoft useful for this kind of application several ad hoc features are included in the concept of FunSoft and its implementation in LEU which are not at all well understood from a theoretical point of view.

It is the purpose of this paper to improve this situation not by Petri net technology transfer from theory to practice but by reverse transfer from practice to theory. This means that we consider the main features used explicitly in FunSoft or implicitly in LEU and try to abstract them in a theoretically clean and comprehensive way leading to a generic concept of high-level nets. The features are transfered to theory in order to describe, and examine them independently of any system implementation in a similar way as abstract data types stack, queue, etc. For example, there are implementation dependent aspects like numbering of arcs or referencing of objects in the FunSoft definition. Additionally we want to lift special cases to a more abstract description. The formalization of the features used in practice is the basis for further examination of structuring, rule-based modification, analysing etc., and the interrelation between them. In fact, both, the explicit data model of FunSoft, based on extended entity/relationship (EER) models, and the explicit organizational model used in LEU, based on a simple hierarchical tree, are not essential for the meaning of FunSoft. It is even easier for comprehension to replace the explicit data and organizational models by abstract models including essentially only an interface for possible explicit data and organizational models. In this way we obtain a parameterized version which can be instantiated in different ways for different kinds of applications. Our new concept of parameterized FunSoft nets, called ParFun, is generic with respect to the data and organizational model and includes FunSoft features like guarded transitions, copy arcs and job execution. Other interesting concepts of FunSoft like transition invocation, feedback modification, time and flexible arcs are not included in ParFun, but we discuss possible extensions in this direction perhaps as constructions on top of ParFun nets. In fact, we are convinced that specific features of FunSoft nets, like place fusion and transition substitution, should not be used explicitly in the concept of a net but be superimposed constructions on net classes. For this reason our ParFun approach includes general notions of fusion, union, and rule-based modification defined as structuring and refinement techniques for ParFun nets. These techniques have been studied for algebraic high-level nets in [PER95] and for abstract Petri nets, a categorical approach unifying different kinds of low-level and high-level net classes, in [Pad96]. As main new technical results in this paper we are able to show interesting compatibility results for fusion and union with rule-based modification for ParFun nets. The idea to study those concepts of nets in theory which have been used successfully in practice already can be considered as reverse Petri net technology transfer.

In section 2 of this paper we review the main concepts of FunSoft nets in the application context of workflow management. Our new concept of ParFun nets as a parameterized abstraction of FunSoft is formally presented in section 3. We also show how to instantiate ParFun to recover the essential parts of FunSoft, how to obtain interesting new variants for other applications, and how different views of ParFun are related to other well known net classes. In section 4 we discuss ParFun as an approach on the boundary of theory and

applications. On a conceptual level we present the techniques of union, fusion and rule-based modification for ParFun nets and corresponding compatibility results, which are important for practical applications. A formal version of these techniques and results and the main ideas of the proofs, which are based on category theory, are given in our appendix. On the other hand we discuss how those additional features of FunSoft, which are available in LEU, could be realized as extensions of our ParFun approach. This would keep the practical relevance for the applications within workflow management and to open it up for similar new kinds of applications. The idea of reverse Petri net technology transfer is also one of the main aims of our new DFG-Research Group "Petri Net Technology" [WRE95], where a subsystem of WIS is one of three main case studies. In the conclusion we summarize the main new concepts and results of this paper.

2 A Petri Net Technique for Workflow Management

The importance of workflow management also known as business process (re)engineering has constantly increased during the past few years. Workflow management deals with modelling, analysis and execution of workflow processes which are sets of logically connected activities pursuing a commercial object. Many applications of workflow management are aiming at commercial and administrative business processes, that are usually strongly structured and with a great potential of computerization. There are numerous languages for modelling business processes, which gradually put more emphasis on a graphical representation of such processes. Here Petri nets become more and more prominent as they visualize processes very intuitively (see [DO96]).

Modelling of Workflow Management with FunSoft

A very notable approach to workflow management is the FunSoft approach (see [Gru91, Gru96]) which forms the basis of the commercial workflow management system LEU [GW95]. This system has been applied with great success to several fields of workflow management with a special focus on a housing management system WIS (see [GGK96]).

According to the requirements of workflow management the FunSoft approach includes modelling, analysis and execution of workflow processes. Models are not only created and analyzed, but also serve as the basis of the enactment of such processes. This means that functions (or subroutines) are called and operate on data whenever an activity of the process is performed. For each of the interacting components — humans, data and processes — there are different models corresponding to different views of the system:

- On the highest level there is the *organizational model*. Here the structure of the company is captured, that is the persons and their competences. In general, this structure is hierarchically ordered and the competences are actualized by *roles.*
  For this purpose trees are used in FunSoft.
- In the *process model* there are the relevant activities and states of the process. Moreover, an appropriate order of the activities is described.
  In the FunSoft approach a complete version of high-level Petri nets is used in the process model.
- The structure of the processed data as well as their relations is captured by the *data model.*
  Extended entity/relationship diagrams (EER-diagrams) (see [Gog94]) are used as data model in FunSoft.

Concepts of FunSoft

The FunSoft approach considered in this paper is the notion of FunSoft nets in [Gru91] and the way they are used in the system LEU [GW95]. In FunSoft the process and the data model are linked together by typing of the places (in FunSoft: channels). The integration of process and organizational model is performed through "manual" transitions (in FunSoft: interaction agencies), to which roles are assigned. Roles are a set of competences which can be either directly linked to a person or to a position within the company. Figure 1, which sketches the relationship between the different models, is taken from [Gru96].

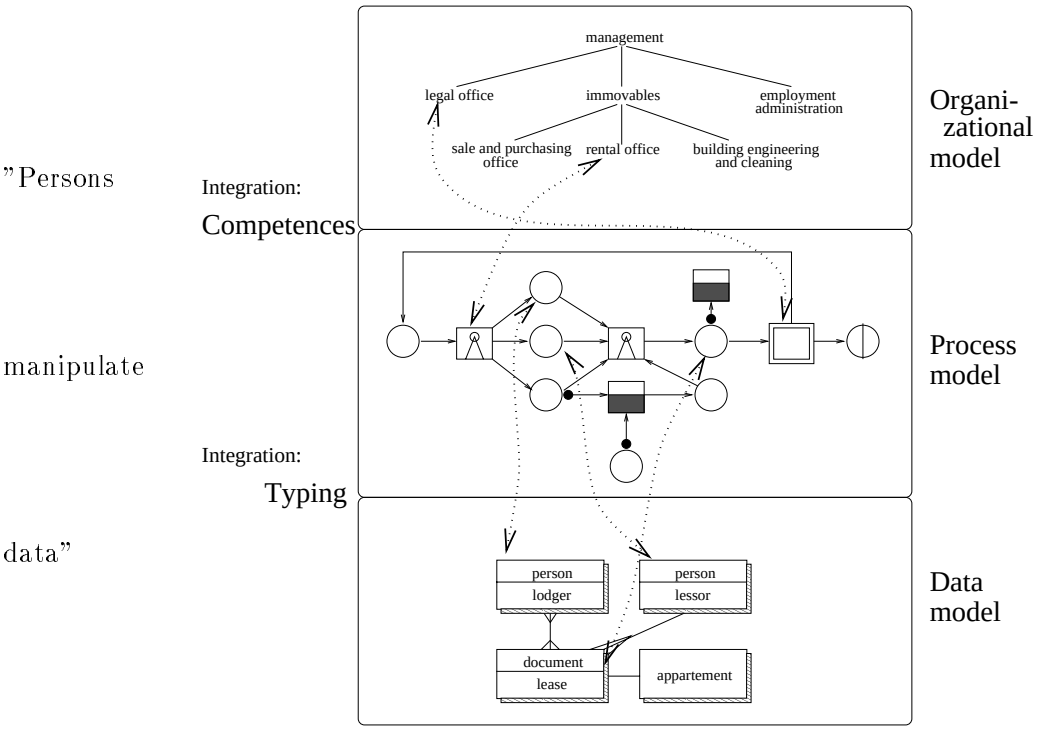


Figure 1: Overview over relationships between models

Before we go into details of the process model, we first sketch analysis and execution[1]. Analysis is mainly done by simulation, where black tokens or simple object identifiers are used in contrast to real data. This abstraction causes the need of manual interruption in case of decisions, which are determined by values. Another possibility to regain a faithful simulation is by assigning probabilities to transitions, such that conflicts can be resolved automatically. On the whole the effects of firing of transitions are imitated in order to get an idea how the model works. Simulation can be done in a partly undefined model, whereas another technique of analysis, namely execution protocols, can only be used in a completely specified model. This, of course, yields more reliable statements about the system than mere simulation.

The model is ready to be executed as soon as all data types, functions, refinements etc. are fully specified. Then a process interpreter controls the enaction, performing several steps. First he locates activated transitions which are then distinguished into manual and automatic ones. In case of an interactive transition the interpreter offers the corresponding action to competent

[1] These can only be explained with regard to the tool LEU and not by the pure definition of FunSoft.

persons (according to their role) via an agenda. In the other case the corresponding subroutine is executed automatically. Furthermore, tokens are shifted by the process interpreter, such that the underlying net is completely invisible to the users.

The following concepts of high-level nets explained in detail in section 3 and 4 are characteristic for FunSoft nets as used in LEU:

> As mentioned above, there is a data model based on EER-diagrams with certain object types and an organizational model based on trees. Transitions are guarded, they can be substituted or invoked as a means of refinement, and jobs can be executed whenever the transition fires. For simulation purposes firing consumes some time. Refinement mechanisms like place fusion and feedback modification (not realized in LEU) are built-in in the notion of nets. There are copy arcs and flexible arcs which allow a dynamic change of weight. Last but not least different access strategies to tokens on places can be considered.

In the FunSoft net shown in the middle part of figure 1 some of the concepts of this approach are depicted. Places are labelled with an object type. Some of them are fusion places ⨁. There are different icons for transitions: for interactive transitions,  for automatic transitions and  for refined transition. Arcs with a dot at the beginning  mean copying — in contrast to consuming — of token.

From FunSoft to ParFun

We have already pointed out that FunSoft covers a lot of modelling concepts, which are useful especially for workflow modelling, for example concepts for modelling data and organizational aspects. In the workflow management system LEU, based on FunSoft, the data aspects are realized by EER models, and the organizational aspects are realized as roles represented by a tree structure. Of course, it is important in order to be more flexible for practical applications to consider also other techniques for the data as well as the organizational model. The data model, for example, could also be presented by an object-oriented model and the organizational model could be nonhierarchical. However, it is difficult to exchange the modelling techniques, as they are tightly integrated into FunSoft and LEU. A flexible use of developing systems like LEU can be obtained, if the data and organizational models are parameterized[2]. This is one of the main reasons to develop in this paper a parameterized abstraction of FunSoft, called ParFun. With ParFun we want to give a framework for workflow oriented systems, that supports the exchange of modelling techniques and the modification of model integration. That is, that instantiation of a formal parameter works on two levels: for a concrete model given in a concrete specification technique one first has to transform the technique and then transform the model in such a way, that it becomes one of the transformed technique[3].

Defining a suitable data and organizational interface, the change from one modelling technique to another is reduced to the change in the implementation of the model, where the interface is unchanged. Figure 2 depicts this approach as realized in ParFun.

In addition to this important parametrization aspect our intention with ParFun is to present — at least for a suitable kernel of FunSoft — a precise technical version for this kind of high-level nets which is easy to understand and to handle from a theoretical point of view.

[2] A further parameter for aspects of time does not seem to be adequate due to the incompatible concepts of time.

[3] We are grateful to Ekkart Kindler for several fruitful discussions.

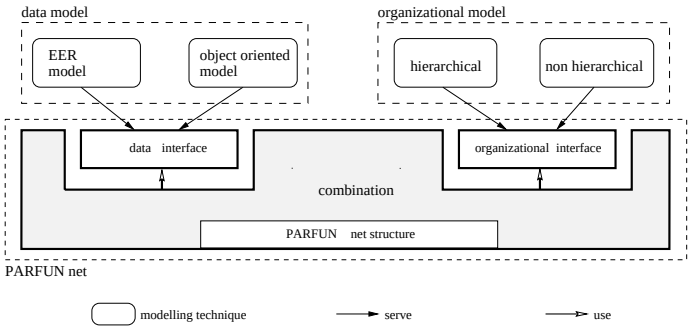


Figure 2: ParFun: A framework for the integration of the different workflow models

Especially we are interested in having structuring techniques — like fusion, union and rule-based modification — for workflow management applications based on ParFun instead of FunSoft. In fact, we are able to provide these structuring techniques and important compatibility results for ParFun in this paper. In the case of algebraic high-level nets these techniques and the corresponding compatibility results have turned out to be most important in a case study for requirement engineering of a medical information system (see [EPE96]).

3 ParFun: A Parameterized Abstraction of FunSoft

In this section we present the main concepts of ParFun and a formal version of ParFun nets including enabling and firing. The interpretation of ParFun nets in different modes allows establishing a clear relationship to place/transition nets, different types of high-level nets and to FunSoft nets, respectively. These interpretations depend on different instantiations of the parameters of ParFun nets. These permit on the one hand the recovery of the essence of FunSoft and on the other hand to formulate new net types which are interesting for further applications. In this sense ParFun can be considered as a parameterized abstraction of FunSoft.

Concepts of ParFun

One of the most important concepts of ParFun is that of *parameters* allowing different parameter instantiations.

Concept 3.1 (ParFun **Parameters and Nets**)
Parameters in the definition of ParFun nets are the

- *data model* and the
- *organizational model*

This leads to a generic definition of nets where the parameters can be instantiated in different ways leading to different actual net types. Figure 3 sketches the relation between ParFun as a parameterized net class and its instances. These instances are obtained by actualization of the data model and the organizational model.
ParFun is a parameterized net class in the sense of [EP97] which describes a family of net classes varying over the different actualizations of its parameters. Thus, there is a conceptual

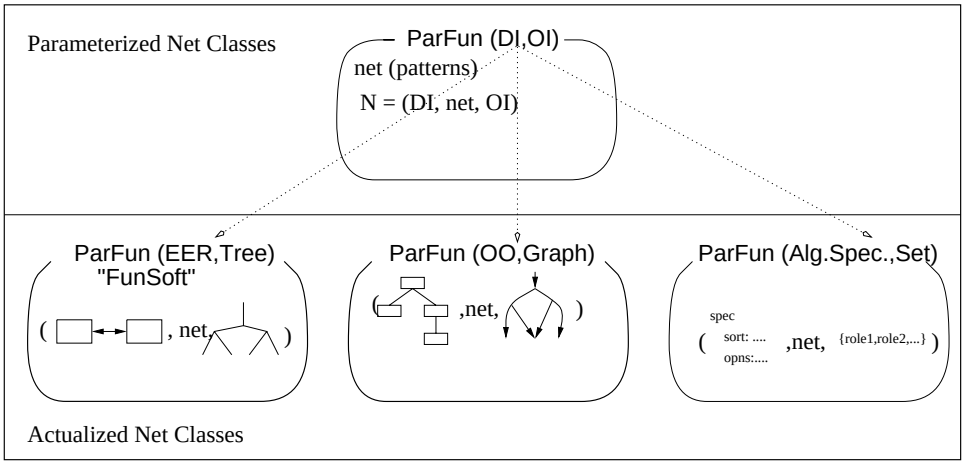


Figure 3: ParFun and its instances

difference between ParFun nets (called net patterns in [EP97]) and nets of some ParFun instance. The generic version of ParFun (and of parameterized net classes in general) does not yield concrete nets, as the formal parameters are not yet specified, they give rise to net patterns. Net patterns constitute a pattern for Petri nets consisting of places, transitions, arcs and functions for the integration of the formal parameters. Nevertheless, neither the kind of specification and its model, nor the organizational model are fixed, due to the missing actual parameters. It should be pointed out that enabling and firing is defined already for the generic version, i.e. ParFun net patterns.
As the focus of this paper is on ParFun, and thus neither on the nets of its instances nor on the difference between these nets and net patterns, we continue using the term ParFun nets.

◊

Next we consider important concepts of FunSoft which are realized in ParFun nets.

Concept 3.2 (ParFun Net)
ParFun nets are high-level Petri nets with parametererized data and organizational model and the following concepts:

- *guarded transitions*, as known from predicate-transition nets
- *copy arcs* allowing *read access* to tokens. This means tokens have to be present for firing, but they are not consumed, as for example in the notion of contextual nets in the sense of Montanari and Rossi in [MR94].
- *job execution* allows executing jobs associated to a transition, where the result of the job is given as a separate output of firing a transition which may or may not influence tokens of the postdomain.
- *access strategies*, like *lifo, fifo, random.* This permits firing transitions with different access strategies for the data, which is considered in our notion of ParFun nets with access strategies 4.6.

◊

Finally we give an overview of the main concepts of the ParFun approach, to be presented in the next section, where only specific aspects are realized within FunSoft.

Concept 3.3 (ParFun Approach)
The ParFun approach includes the following concepts which are not part of ParFun, but are defined on top of ParFun nets:

- *Interpretation modes* are modes which allow different ways of assigning values to tokens leading to a low-level, high-level, and a role specific view of ParFun nets.
- *Fusion* allows the fusion of subnets. This generalizes the concepts of place fusion, realized in FunSoft, and transition fusion well known from literature.
- *Union* allows constructing the union of two (or more) nets with shared subnets, known for algebraic high-level nets [PER95].
- *Rule-based modification* generalizes different refinement concepts known in literature, where a rule specifies those subnets which have to be removed and added, respectively. This technique is well-known in the theory of graph grammars and has been adapted to algebraic high-level nets in [PER95] and to abstract Petri nets in [Pad96]. It includes as a special case the concept of transition substitution in FunSoft.
- *Net morphisms* are morphisms between ParFun nets, which allow expressing the structural relationship between nets, especially within the formal definition of fusion, union and rule-based modification (see appendix). Net morphisms are known for several types of nets in literature.

◊

Formal version of ParFun

In this subsection we introduce the mathematical formulation of ParFun according to the concepts discussed in 3.2.

Definition 3.4 (ParFun Parameters)
The ParFun parameters are given by a pair (D, R), where D denotes the data type parameter and R denotes the organizational parameter:

1. $D = (SIG_{PF}, A, F, \models)$ is a data type with:
   - $SIG_{PF} = (Types, X, J)$ a signature consisting of:
     $Types$ is a set, whose elements are called types.
     $X = (X_t)_{t \in Types}$ is a family of typed variables.
     J is a set of function symbols $j : v \to w$, for short jobs, with $v, w \in Types^*$. [4]
   - A model A in the model class $Mod(SIG_{PF})$, where SIG_{PF} is considered as a signature without variables.
   - A set of formulas F over the signature SIG_{PF} with variables X.
   - A satisfaction relation
     $\models \subseteq Mod(SIG_{PF}) \times F$ such that $A \models \varphi$ iff A satisfies the formula φ.
2. R is a set, whose elements are called roles.

□

[4] For a given set X we use the operator "X^*" as Kleene star yielding strings over the alphabet X. In contrast "X^+" designates non-empty strings. "$X^{\circledast}$" is the free commutative monoid over X.

Definition 3.5 (ParFun Net)
A ParFun net PN is given by $PN = (D, R, N, C, M_0)$ where

- $D = (SIG_{PF}, A, F, \models)$ is a ParFun data type with $SIG_{PF} = (Types, X, J)$.
- R is a set of roles
- $N = (P, T, arc)$ is the underlying net with
  - P is a set of places.
  - T is a set of transitions.
  - $arc = \{pre, post, copy : T \to (X \times P)^{\circledast}\}$ is a set of functions, where $pre(t), post(t)$ and $copy(t)$ for each $t \in T$ are the pre, post and copy domains ot t, which can be represented as finite formal sums $\sum(x_i, p_i)$ with $x_i \in X$ and $p_i \in P$ $(i = \{1, \ldots, n\})$ meaning that the arc between t and p_i is labelled by x_i.
- $C = (sort, job, cond, role)$ is a family of functions defining the combination of D, R and N.
  - $sort : P \to Types$ is the typing of places.
  - $job : T \to \mathcal{P}(J)$ assigns a set of jobs to transitions.
  - $cond : T \to \mathcal{P}(F)$ assigns a set of conditions to transitions.
  - $role : T \to \mathcal{P}(R)$ assigns a set of roles to transitions.
- $M_0 \in (A_{sort(p)} \times P)^{\circledast}$ is the initial marking[5].

□

The next step is to present the firing of ParFun nets. Note, that we do not require parameters to be actualized, but can already formulate the dynamic behaviour for ParFun nets. As nets of actualized ParFun instances can be considered to be a variant of high-level nets, the enabling and firing of a transition is closely related to the corresponding concepts in various high-level nets. Similarily, we use the assignment of variables to describe the involved tokens and to check the transition conditions. An assignment is a family of functions $asg = (asg_{typ} : X_{typ} \to A_{typ})_{typ \in Types}$, Subsequently, we use $asg : X \to A$ for short. Given an assignment $asg : X \to A$ and a formula φ. $asg(\varphi)$ assigns values of the model A to every free variable in φ.

Definition 3.6 (Enabling and Firing of Transitions)
Let $t \in T$ be a transition, $asg : X \to A$ an assignment, $M \in (A_{sort(p)} \times P)^{\circledast}$ a marking and $r \in R$ a role. t is enabled under the assignment asg in the marking M for the role r, if and only if the following conditions hold:

1. Valid Assignment
   $A \models asg(cond(t))$ and $r \in role(t)$
2. Enabling
   $M \geq pre_{asg}(t) + copy_{asg}(t)$ where
   $pre_{asg} : T \to (A_{sort(p)} \times P)^{\circledast}$ is defined as $pre_{asg}(t) = \sum(asg(x_i), p_i)$ if $pre(t) = \sum(x_i, p_i)$
   and analogously
   $copy_{asg} : T \to (A_{sort(p)} \times P)^{\circledast}$ is defined as $copy_{asg}(t) = \sum(asg(x_i), p_i)$
   if $copy(t) = \sum(x_i, p_i)$

[5]The index $sort(p)$ of A in $(A_{sort(p)} \times P)$ excludes pairs (a, p) with $a \notin A_{sort(p)}$.

3. Firing
   Let $t \in T$ be enabled under the assignment $asg : X \to A$ in the marking $M \in (A_{sort(p)} \times P)^{\circledast}$. The firing of t leads to the follower marking M':

$$M' = M - pre_{asg}(t) + post_{asg}(t)$$

   and to output $asg(job(t))$, which means output values $asg(j)$ of the model A for each $j \in job(t)$.

□

Corresponding to simulation and execution of a FunSoft net we present different modes of interpretation of a ParFun net. These modes are independent of the definition of firing, but are concerned with the availability of the data and organizational model. If both are not specified the interpretation can yield a simple simulation, only using black tokens, thus simulating merely the process view. If only the data model is available, the interpretation leads to a simulation that includes the handling of data. Nevertheless, for manual transitions jobs may not be defined sufficiently. These are handled by assigning values arbitrarily.

Concept 3.7 (Interpretation Modes)
There are three interpretation modes which differ in the way of assigning values to tokens and in taking the organizational structure into account. In the interpretation mode

1. simple tokens get trivial values implying that they are undistinguishable. Furthermore roles are neglected in this mode. $A = \{*\}$ as a model of the data type and $R = \{*\}$ the trivial role correspond to this mode. This firing behaviour corresponds to simulation in FunSoft and is achieved by firing transitions with black tokens. All firing conditions are satisfied trivially.

2. simulate arbitrary values are assigned to tokens. Roles are neglected as above. This means that the model A is the one given in the actualized net and $R = \{*\}$ is the trivial role.

3. enaction values of the data type are assigned interactively by humans. The assignment can only be carried through by competent persons (according to their role). This mode captures the enaction of FunSoft nets, where the interface to the environment is essential.

◊

The mode simple relates ParFun nets to the underlying low-level net. This construction corresponds to the skeleton of a high-level net. The mode simulate corresponds to the firing behaviour of high-level nets known from literature.

Instantiations of ParFun

In this subsection we discuss the possible instantiations of the ParFun parameters. These instantiations are merely sketched and are based on results known from literature. The parameterization of ParFun is considered to be a static one, meaning the data model is assumed to be given. Thus the matching of the actual parameter is not required to take any dynamic semantics of the model into account. Figure 4 sketches a small example of a net of the ParFun instance actualized with EER and trees. This is the variant that corresponds to FunSoft.

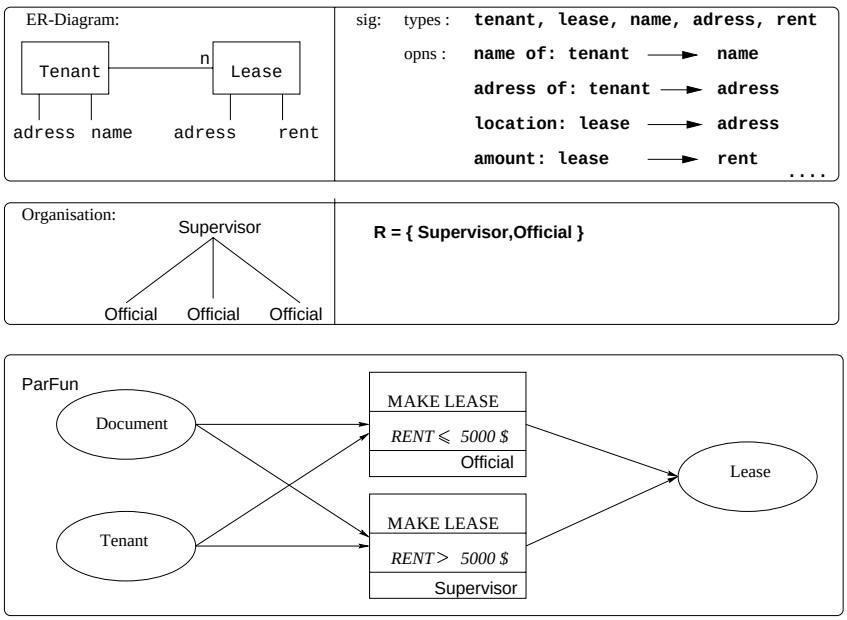


Figure 4: Example

Concept 3.8 (Instantiation with Extended Entity/Relationship Models)
Here we refer to the well known approach to relate entity/relationship schemes to abstract data types. Based on extended entity/relationship (EER) models (see also [Gog94]) consisting of arbitrary data types for attribute domains, optional and multivalued attributes, and complex structured entity types, we employ the main result in [GH91] concerning the relation to abstract data types. Another more pragmatic approach to relating entity/relationship models to abstract data types is given in [Het93], where entity/relationship models are translated into SPECTRUM, an algebraic specification language. In both cases there is a transformation from the entity/relationship model to a signature and a suitable model of the signature. Both are used in the data type of ParFun. Moreover, the types of this signature can be used together with a set of predicates to construct first-order predicate formulas over the signature.

◊

Concept 3.9 (Instantiation with Object Oriented Data Models)
Object orientation becomes more and more important for software development as well as for business process management. Therefore, it is interesting to instantiate ParFun with an object oriented data model (class diagram). There are several object oriented techniques – like OMT [RBPEL91] – providing their own graphical conventions for the denotation of the several concepts like object identity, classes, generalization, aggregation and associations. Thus, any possible object model built by these various concepts can be supplied with these conventions. An approach that presents a mathematical foundation for a large class of object models — not necessarily denotated in OMT — can be found in [CLWW95]. This approach supports any graphical formalisms. The static semantics for object oriented structures are modelled by suitable signatures, so called attributed graph signatures with contraints. These signatures (SIG_{OO}) contain the signature for the graphical part, the data part, the attribute part, and axioms (a set of first order formulas). For the instantiation of the ParFun data part with an object oriented data model we can use the above approach. The ParFun signature SIG_{PF} is then given by the data part of the signature SIG_{OO}. Moreover, the carrier sets of the model concerning the data part of the signature SIG_{OO} can be used for the model required in the

ParFun parameter. Similarily, the jobs are given by the operations specified in the data part of the signature SIG_{OO}. ◊

Concept 3.10 (Instantiations of the Organizational Model)
The organizational model used in the FunSoft approach, especially in the workflow management system LEU, is a hierarchy of roles modelled by a tree. For the formalization within ParFun nets it is only essential to have a set R of roles and an integrating function *role* which assigns a set of roles to each transition. Hence, the instantiation of R by the carrier set of a tree allows recovering the organizational model of FunSoft. Of course, it makes also sense to consider on one hand a more restricted linear hierarchy modelled by a tree without branching, or on the other hand a more liberal hierarchy allowing multiple predecessors, which can be modelled by a collapsed tree or a rooted directed acyclic graph. Moreover, the organizational model may have some roles which are not connected to each other or even with cyclic dependencies such that it should be modelled by a forest, an acyclic graph or an arbitrary graph. This leads to other meaningful instantiations by different kinds of trees and graphs. ◊

4 ParFun on the Boundary of Theory and Applications

In this section we define structuring and refinement concepts of ParFun nets which are essential for all kinds of practical applications. Similar to algebraic high-level nets in [PER95], and abstract Petri nets in [Pad96] we define fusion and union as horizontal structuring techniques and rule-based modification as vertical refinement technique. Our main technical results show under which conditions fusion and union of ParFun nets are compatible with rule-based modification. In fact, the FunSoft concepts of place fusion and transition substitution are special cases of fusion and rule-based modification for ParFun nets and hence covered in our ParFun approach. In order to reach the full range of applications, that are possible for FunSoft, for ParFun as well we discuss how the missing concepts of FunSoft can be introduced as extensions of ParFun. For this purpose we extend ParFun by access strategies and discuss briefly how the remaining missing concepts might be handled by extensions of the ParFun approach. On the other hand our parameterization concept and the general notions of fusion, union and rule-based modification within the ParFun approach are going beyond the modelling power of FunSoft and our main results are important from a theoretical as well as a practical point of view. It remains open how far these results remain valid for the practical extension of ParFun discussed above. In this sense ParFun is really on the boundary of theory and applications.

Structuring Techniques for ParFun Nets

In the following we present conceptual definitions and results for fusion, union and rule-based modification. A short technical version of these concepts, results and the corresponding proofs are given in the appendix. Usually the concept of fusion in literature (see e.g. [Jen92]) means either place fusion or transition fusion. We allow fusion and union of nets w.r.t. arbitrary subnets. A precise notion of subnet is given in the appendix. Note, that these structuring techniques are static ones, that is they operate on nets without initial marking, which will be called "net structures" subsequently.

Definition 4.1 (Fusion and Union of ParFun Net Structures)

- Given a ParFun net structure P containing two copies of a subnet F we obtain the resulting ParFun net structure P' by fusion of the two copies of F. In this case P' is called *fusion* of P by F, written $P \overset{F}{\circ\!\!\longrightarrow} P'$ (see also appendix, definition A.3).

- Given two ParFun net structures P_1 and P_2 with a shared subnet I, called *interface* then the new net structure P obtained by a union of P_1 and P_2, which is disjoint except of the shared subnet I, is called *union* of P_1 and P_2 via I, written $(P_1, P_2) \overset{I}{\rightarrowtail\!\!\!\Rightarrow} P$. (see also appendix, definition A.3).

□

The idea of rule-based modification of net structures is to define rules $p = (L \leftarrow K \rightarrow R)$ consisting of net structures L, K, R which can be applied to a net structure P in such a way that L is replaced by R with interface K leading to a transformation $P \overset{p}{\Longrightarrow} Q$ from net structure P to net structure Q via rule p (see [PER95, Pad96] for rule-based modification of algebraic-high-level and abstract Petri nets).

Definition 4.2 (Rule-Based Modification of ParFun Net Structures)

- A rule $p = (L \leftarrow K \rightarrow R)$ consists of ParFun net structures L, K, R where K is a shared subnet of L and R, called *left and right hand side* of p, respectively.
- A rule $p = (L \leftarrow K \rightarrow R)$ can be applied to a ParFun net structure P at a subnet L' of P, if L', K' are isomorphic copies of L, K and removing $L' - K'$ from P leads to well-defined ParFun net structure D with subnet K'.
- Given a rule $p = (L \leftarrow K \rightarrow R)$ which can be applied to a ParFun net structure P at L' with subnet K' we obtain a new ParFun net structure Q in two steps:
  - REMOVE: We remove $L' - K'$ from P leading to D with subnet K'.
  - ADD: We add an isomorphic copy R' of R with subnet K' to D leading to a net structure Q as the result of the union construction $(D, R') \overset{K'}{\rightarrowtail\!\!\!\Rightarrow} Q$.

  In this case we obtain a *transformation* from ParFun net structure P to ParFun net structure Q via rule p, written $P \overset{p}{\Longrightarrow} Q$.

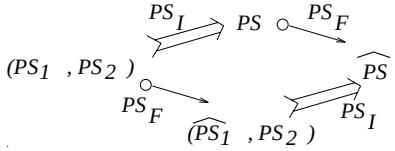
(See also appendix Def. A.5 and Char. A.6) □

Note, that the FunSoft concept transition substitution is a special case of rule-based modification, where the left hand side L of the rule is a single transition.

Compatibility Results

Now we are able to formulate the main results concerning compatibility of fusion and union with rule-based modification.

Theorem 4.3 (Compatibility of Union and Fusion)
Given a union $(P_1, P_2) \overset{P_I}{\rightarrowtail\!\!\!\Rightarrow} P$ and a fusion $P_1 \overset{P_F}{\circ\!\!\!\rightarrow} \widehat{P_1}$, then we can apply fusion and union in any order, that is:



□

Theorem 4.4 (Compatibility of Fusion and Rule-Based Modification)
Given a fusion $P \circ\!\!\xrightarrow{F} P'$ and a transformation $P \stackrel{p}{\Longrightarrow} Q$ there is an induced fusion $Q \circ\!\!\xrightarrow{F} Q'$ and an induced transformation $P' \stackrel{p}{\Longrightarrow} Q'$, provided that the fusion is independent of the transformation in the sense that F is preserved by the transformation:

$$P \stackrel{p}{\Longrightarrow} Q \circ\!\!\xrightarrow{F} Q' = P \circ\!\!\xrightarrow{F} P' \stackrel{p}{\Longrightarrow} Q'$$

(See also appendix, Thm. A.8) □

Theorem 4.5 (Compatibility of Union and Rule-Based Modification)
Given a union $(P_1, P_2) \stackrel{I}{\rightarrowtail} P$ and transformations $P_1 \stackrel{p_1}{\Longrightarrow} Q_1$ and $P_2 \stackrel{p_2}{\Longrightarrow} Q_2$ then there is an induced union $(Q_1, Q_2) \stackrel{I}{\rightarrowtail} Q$ and an induced transformation $P \stackrel{p_1+p_2}{\Longrightarrow} Q$ with the parallel rule $p_1 + p_2 = (L_1 + L_2 \leftarrow K_1 + K_2 \rightarrow R_1 + R_2$ and the same ParFun net structure Q, provided that the union is independent of the given transformation in the sense that I is preserved by the transformation.

$$(P_1, P_2) \stackrel{I}{\rightarrowtail} P \stackrel{p_1+p_2}{\Longrightarrow} Q = (P_1, P_2) \stackrel{p_1+p_2}{\Longrightarrow} (Q_1, Q_2) \stackrel{I}{\rightarrowtail} Q$$

(See also appendix, Thm. A.10) □

ParFun Nets with Access Strategies

One important concept of FunSoft is the possibility to fire transitions with different access strategies for the data, like *lifo, fifo, random*. In the following, we show how to extend ParFun nets by the concept of access strategies.

Definition 4.6 (ParFun Net with Access Strategies)

- A ParFun net with access strategy $PN_{access} = (D, R, N, C, M_a, access)$ is a ParFun net structure $PN = (D, R, N, C)$ with
  - for each transition t we assume that $pre(t), post(t), copy(t)$ are partial functions $f : P \to X$
  - the markings M are functions $M : P \to A^*{}_{sort(p)}$
  - $access$ is an additional (higher order) function $access : P \to \{lifo, fifo, random\}$
- The access functions $lifo, fifo, random : A^*{}_{sort(p)} \to \mathcal{P}(A_{sort(p)})$ are defined by
  - $lifo(\lambda) = fifo(\lambda) = random(\lambda) = \emptyset$ for the empty string λ
  - $lifo(w) = \{a_n\}, fifo(w) = \{a_1\}$ and $random(w) = \{a_1, \ldots, a_n\}$ for $w = a_1 a_2 \ldots a_n$
- A transition t in PN_{access} is enabled under an assignment $asg : X \to A$, a role $r \in R$, and a marking M if we have
  - A satisfies $asg(cond(t))$
  - $r \in role(t)$
  - $asg(pre(t)(p)) \in access(p)(M(p))$
  - $asg(copy(t)(p)) \in access(p)(M(p))$
- Let transition t in P_{access} be enabled under an assignment asg, a role $r \in R$ and a marking M, then the firing of t leads in a first step to a marking M' defined for all p in the domain of $pre(t)$ by

- $lifo : M'(p) = front(M(p))$
- $fifo : M'(p) = tail(M(p))$
- $random : M'(p) = M(p) - asg(pre(t)(p))$, where "$-$" deletes an arbitrary occurrence of $asg(pre(t)(p))$ in the list $M(p)$

and $M'(p) = M(p)$ for all other $p \in P$. In a second step we obtain a marking M'' defined for all p in the domain of $post(t)$ by

- $M''(p) = radd(M'(p), asg(post(t)(p))$, where $radd$ is the addition of an element to a list from the right hand side.

and $M''(p) = M'(p)$ for all other $p \in P$.
Furthermore, $asg(job(t))$ is output, which means output values $asg(j)$ of the model A for each $j \in job(t)$.

□

ParFun in Contrast to FunSoft

In the following we discuss all further FunSoft concepts which are not explicitly included into our notion of ParFun nets up to now. Those concepts which are useful from a practical point of view and should also be considered in extended versions of ParFun or the ParFun approach consisting of ParFun nets together with structuring techniques, e.g. union, fusion and rule-based modification (see 3.3).

Concept 4.7 (Place Fusion)
The FunSoft concept of place fusion is a special case of fusion defined for ParFun net structures above where the subnet F consists of places only. Since fusion is a horizontal structuring technique we prefer to have it in the ParFun approach rather as part of the definition of nets.
◊

Concept 4.8 (Transition Substitution)
The FunSoft concept of transition substitution is a special case of rule-based modification where the left-hand side of the rule consists of a single transition (possibly with adjacent places) only. Since rule-based modification is a vertical refinement technique we prefer to have it in the ParFun approach rather as part of the definition of nets. ◊

Concept 4.9 (Transition Invocation)
While transition substitution changes the structure of the net in a static way independent of the steps of an execution the concept of transition invocation changes the structure of the net in a dynamic way. Similar to the concept of procedure calls each invocation generates a new copy of this subnet and execution of the main net can only be continued if the execution of the subnet has terminated. In this case the corresponding copy of the subnet is removed. A similar concept has already been considered in [HJS90]. In order to include it into the ParFun approach we would have to define a hybrid execution formalism including firing transitions and transformations defined by rule-based modification as basic execution steps. ◊

Concept 4.10 (Flexible Arcs)
Firing of transitions in FunSoft nets can be done in different modes. The all-mode corresponds to the usual case that tokens are removed from all input places of a transition and added to all output places. In the some-mode tokens are added only to some output places according

to a nondeterministic choice. In the det-mode tokens are added only to one of the two output places. The mult-mode allows putting n tokens on one output place if the value n has been generated on a specific other output place. Finally, the complex-mode allows a mixture of the other modes, described by a logic formula. These different firing modes can be considered in some sense as flexible arcs of a transition. In a specific firing mode for a specific marking arcs can be considered to be removed or weighted by 0, if the marking of the corresponding input or output places remains unchanged. The weights of other arcs are changed. A general concept of flexible arcs is included in the notion of algebraic system nets of [KR96] which is realized by multiset-variables. More or less, this corresponds to the idea of allowing the data type A of a net to have relational (partial and nondeterministic) operations instead of total ones. This could also be considered for ParFun nets. ◊

Concept 4.11 (Time)
FunSoft nets allow allocating to each transition a non-negative real number or probabilities corresponding to time consumption during simulation of the net. This permits the calculation of the simulation time for execution sequences of the net or to provide stochastic analysis. A similar concept might also be added to ParFun. ◊

Concept 4.12 (Feedback Modification)
The feedback modification concept of FunSoft allows modifying the net structure during execution. In contrast to transition invocation, firing of a transition t may cause a modification of the net at some other place or transition in the net not adjacent to t and this modification remains also after termination of the execution. However, this concept of feedback modification has turned out to be problematic for practical use, thus it is not realized in the system LEU. Hence, there is no practical need to include it into the ParFun approach. ◊

5 Conclusion

The idea of reverse Petri net technology transfer is to obtain – based on concepts of practical relevance – improved concepts and results in the theory of Petri nets. In this paper we have considered FunSoft, which has turned out to be a very powerful Petri net technique in the area of workflow management. The fact that FunSoft is not well understood from a theoretical point of view has been the motivation to formalize the main concepts of FunSoft in this paper leading to the ParFun approach, a parameterized abstraction of FunSoft. It is important to note, that we distinguish between generic parameters and concepts of ParFun nets and additional concepts superimposed on nets which are in the ParFun approach. Especially, we have decided that the concepts of place fusion and transition substitution of FunSoft nets should not become concepts of ParFun nets but should be considered in a more general form as concepts fusion and rule-based modification in the ParFun approach. Figure 5 shows that the explicit data and organizational model of FunSoft are replaced by generic parameters in ParFun. The essential concepts of FunSoft have been formalized in ParFun, some additional concepts of FunSoft are discussed as possible extensions of ParFun, while feedback modification has turned out not to be useful in practice. Finally, we have interesting additional concepts in the ParFun approach which are not in FunSoft. Future work comprises the formulation of transition invocation, access strategies, time and flexible arcs. Due to the extensive use of transition invocation in our case study WIS we concentrate on the integration of this concept into ParFun. This further step is necessary in order to start the reconstruction of FunSoft. We are confident that this reconstruction could be a solid basis for a more flexible re-implementation of the FunSoft tool LEU.

Concepts	FunSoft	ParFun
data model	EER	generic
organizational model	tree	generic
guarded transition	+	+
copy arcs	+	+
job execution	+	+
place fusion	+	approach
transition substitution	+	approach
transition invocation	+	?
access strategy	+	?
time	+	?
flexible arcs	+	?
feedback modification	(+)	-
fusion	-	approach
union	-	approach
rule–based modification	-	approach
net morphisms	-	approach
interpretation modes	-	approach

Figure 5: Comparison between FunSoft and ParFun

The benefit of our ParFun approach from the theoretical point of view is the formulation of practically relevant FunSoft concepts and of the structuring techniques fusion, union and rule-based modification based on net morphisms, as well as important compatibility results, stating under which assumptions fusion, union and rule-based modification are pairwise compatible. These structuring techniques are even more important from a practical point of view in order to handle complex systems in an adequate way. Moreover, for flexibility of instantiation the parameters of ParFun allow to extend FunSoft to modelling of workflow management based on object oriented data models and other kinds of organizational models.

Appendix

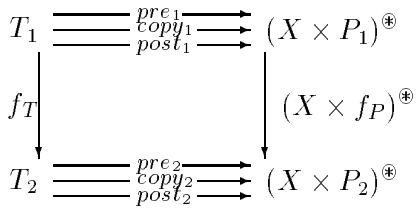
A Technical Notions and Results

In this appendix we give the categorical notions and result underlying the conceptual results in section 4. These concepts generalize to those introduced in section 4 by using general net morphisms instead of subnets corresponding to subnet inclusion.

Fact A.1 (Category PARFUN)
The category **PARFUN** of ParFun net structures with respect to given data types and roles consists of objects $PS = (D, R, N, C)$ which are ParFun nets (see Def. 3.5), without inititial marking, called ParFun net structures, but with fixed data part D and fixed roles R. Net morphisms $f_N : PS_1 \to PS_2$ with $PS_i = (D, R, N_i, C_i)$ are given by $f_N = (f_P, f_T)$, a pair of functions, mapping places to places $f_P : P_1 \to P_2$ and transitions to transitions $f_t : T_1 \to T_2$. The subsequent conditions have to be satisfied:

1. Compatibility with the pre-, copy- and post functions, that is the following diagram[6] commutes separately for pre-, copy- and post functions:

[6] More precisely, the variables are indexed sets, thus the mapping is defined by f_P.



where $(\_)^{\circledast}$ denotes the free commutative monoid.

2. Compatibility with the typing of places: $sort_2 \circ f_P{}^{\circledast} = sort_1$
3. Compatibility with the set of jobs: $job_2 \circ f_T = job_1$
4. The net morphism is compatible with the set of conditions: $cond_2 \circ f_T = cond_1$
5. Compatibility with the set of roles: $role_2 \circ f_T = role_1$

□

Lemma A.2 (Finite Cocompleteness of PARFUN)
The category **PARFUN** is finitely cocomplete. □

Proof Idea *For finite colimit construction it is sufficient to have initial objects and pushouts (by dualization of Thm. 12.4 in [AHS90]):*
The initial object is given by $PS_\emptyset = (D, R, N_\emptyset, C_\emptyset)$ where $N_\emptyset$ consists of empty sets of places and transitions as well as of empty pre-, copy- and post functions, and $C_\emptyset$ consists of empty functions. The pushout $PS_1 \stackrel{f'}{\to} PS_3 \stackrel{g}{\leftarrow} PS_3$ of $PS_1 \stackrel{f}{\leftarrow} PS_0 \stackrel{g}{\to} PS_2$ is constructed componentwise, where $T_3 = T_1 +_{T_0} T_2$ and $P_3 = P_1 +_{P_0} P_2$ both are pushouts in **Sets**, *the category of sets. The functions used in PS_3 are given due to the induced pushout morphism of T_3 or P_3. Moreover, these induced morphisms make sure that f' and g' are well-defined and the universal property of PS_3 holds.* √

The following definition makes Def. 4.1 more precise and more general, by using net morphisms. The use of net morphisms allows an arbitrary embedding instead of the subnet F or I.

Definition A.3 (Fusion, Union)
The fusion of two net morphisms $f1, f2 : PS_F \to PS$ is a ParFun net structure $\widehat{PS}$ together with a net morphism $g : PS \to \widehat{PS}$ defined by the coequalizer $(g : PS \to \widehat{PS}, \widehat{PS})$ of $f1$ and $f2$. The fusion is denoted by $PS \circ\!\!\longrightarrow \widehat{PS}$ via $(PS_F, f1, f2, g)$, short $PS \circ\!\!\stackrel{PS_F}{\longrightarrow} \widehat{PS}$.
The union of a pair of ParFun net structures (PS_1, PS_2) via some interface PS_I with the net morphisms $i1 : PS_I \to PS_1$ and $i2 : PS_I \to PS_2$ is given by the pushout $PS_1 \to PS \leftarrow PS_2$ and denoted by $(PS_1, PS_2) \gtrdot\!\!\Longrightarrow PS$ via $(PS_I, i1, i2)$, short $(PS_1, PS_2) \stackrel{PS_I}{\gtrdot\!\!\Longrightarrow} PS$. □

Theorem A.4 (Compatibility of Union and Fusion)
Given a union $(PS_1, PS_2) \stackrel{PS_I}{\gtrdot\!\!\Longrightarrow} PS$ and a fusion $PS_1 \circ\!\!\stackrel{PS_F}{\longrightarrow} \widehat{PS_1}$, then we can apply fusion and union in any order, that is:
$(PS_1, PS_2) \stackrel{PS_I}{\gtrdot\!\!\Longrightarrow} PS \circ\!\!\stackrel{PS_F}{\longrightarrow} \widehat{PS} = (PS_1, PS_2) \circ\!\!\stackrel{PS_F}{\longrightarrow} (\widehat{PS_1}, PS_2) \stackrel{PS_I}{\gtrdot\!\!\Longrightarrow} \widehat{PS}$ □

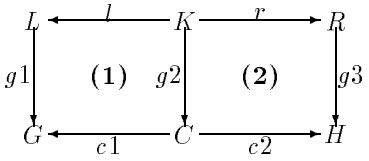
Proof Idea *The underlying constructions of coequalizer and pushout are commutative – due to a nontrivial result from category theory – the commutativity of colimits (see [EM90] page 398) and the fact that both constructions are special colimits.* √

In this section we introduce the concepts known from high-level replacement systems in the sense of [EHKP91]. In high-level replacement systems these concepts are given purely categorically, but here we adapt them to ParFun net structures.

Definition A.5 (Rules and Transformations)

1. A rule $p = (L \stackrel{l}{\leftarrow} K \stackrel{r}{\rightarrow} R)$ in **PARFUN** consists of the ParFun net structures L, K and R, called left hand side, interface and right hand side respectively, and two net morphisms $K \stackrel{l}{\rightarrow} L$ and $K \stackrel{r}{\rightarrow} R$ with both net morphisms $l, r \in \mathcal{M}$, the class of injective net morphisms.

2. Given a rule $p = (L \stackrel{l}{\leftarrow} K \stackrel{r}{\rightarrow} R)$ a direct transformation $G \stackrel{p}{\Longrightarrow} H$, from a ParFun net structure G to an ParFun net structure H is given by the following two pushout diagrams **(1)** and **(2)** in the category **PARFUN** as shown below:



   The net morphisms $L \stackrel{g1}{\rightarrow} G$ and $R \stackrel{g3}{\rightarrow} H$ are called occurrences of L in G and R in H, respectively. By an occurrence of rule $p = (L \stackrel{l}{\leftarrow} K \stackrel{r}{\rightarrow} R)$ in a ParFun net structure G we mean an occurrence of the left hand side L in G.

3. A transformation sequence $G \stackrel{*}{\Longrightarrow} H$, for short transformation, between ParFun net structures G and H means G is isomorphic to H or there is a sequence of $n \geq 1$ direct transformations $G = G0 \stackrel{p1}{\Longrightarrow} G1 \stackrel{p2}{\Longrightarrow} \ldots \stackrel{pn}{\Longrightarrow} Gn = H$. For this sequence we may also write $G \Longrightarrow H$ via $(p1, \ldots, pn)$.

□

The characterization given next corresponds to the two steps REMOVE and ADD in Def. 4.2.

Characterization A.6 (Applicability of Rules)
Given a rule $p = (L \stackrel{l}{\leftarrow} K \stackrel{r}{\rightarrow} R)$, a ParFun net structure G and an occurrence $L \stackrel{g1}{\rightarrow} G$ of L in G, then the rule p is applicable to G via $L \stackrel{g1}{\rightarrow} G$ if and only if the following two conditions are satisfied:

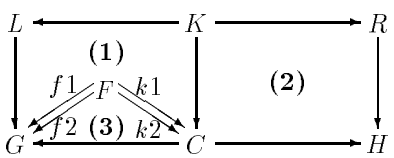
1. There is a ParFun net structure C together with net morphisms $K \stackrel{g2}{\rightarrow} C$ and $C \stackrel{c1}{\rightarrow} G$, such that the square **(1)** in Def. A.5 is a pushout.

2. There is the pushout object H together with net morphisms $R \stackrel{g3}{\rightarrow} H$ and $C \stackrel{c2}{\rightarrow} H$, such that the square **(2)** in Def. A.5 is a pushout.

◊

The following definition – given in Thm. 4.4 merely implicitly – states the assumption required for the compatibility of fusion and rule-based modification.

Definition A.7 (Independence of Fusion and Transformation)
A fusion $G \circ\!\!\longrightarrow G'$ via $(F, f1, f2, g)$ is parallel independent from a transformation $G \stackrel{p}{\Longrightarrow} H$ with $p = (L \leftarrow K \rightarrow R)$ and given by pushouts **(1)** and **(2)**, if there are net morphisms $k1, k2 : F \rightarrow C$ so that the triangle **(3)** commutes componentwise:



□

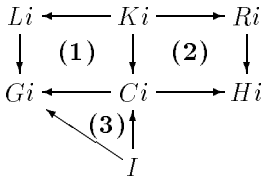
This theorem given in [Pad96] in the more general frame work of high-level replacement systems corresponds to Thm. 4.4 in section 4.

Theorem A.8 (Fusion)
Given a fusion $G \overset{F}{\circ\!\!\longrightarrow} G'$, which is independent from a transformation $G \overset{p}{\Longrightarrow} H$ there is a PAR-FUN net structure H', obtained by fusion $H \overset{F}{\circ\!\!\longrightarrow} H'$ and also by transformation $G' \overset{p}{\Longrightarrow} H'$. That means, we have $G \overset{p}{\Longrightarrow} H \overset{F}{\circ\!\!\longrightarrow} H' = G \overset{F}{\circ\!\!\longrightarrow} G' \overset{p}{\Longrightarrow} H'$. □

Proof Idea *The high-level replacement version of this theorem, given in [Pad96] can be applied to the category* **PARFUN**, *due to finite cocompleteness of the category* **PARFUN**, *see lemma A.2.* √

The following definition – given in Thm. 4.5 merely implicitly – states the assumption required for the compatibility of union and rule-based modification.

Definition A.9 (Independence of Union and Transformation)
A union $(G1, G2) \overset{I}{\rightarrowtail\!\!\Longrightarrow} G$ is called parallel independent from transformations $Gi \overset{pi}{\Longrightarrow} Hi$ for $i \in \{1,2\}$ given by the pushouts **(1)** and **(2)** if there are net morphisms $I \to Ci$ so that the triangle **(3)** commutes for $i \in \{1,2\}$



□

The next theorem given in [Pad96] in the more general frame work of high-level replacement systems corresponds to Thm. 4.5 in section 4.

Theorem A.10 (Union)
Given a union $(G1, G2) \overset{I}{\rightarrowtail\!\!\Longrightarrow} G$ independent from the transformations $Gi \overset{pi}{\Longrightarrow} Hi$ for $i = 1, 2$, then there is an object H obtained by the union $(H1, H2) \overset{I}{\rightarrowtail\!\!\Longrightarrow} H$ and by the transformation $G \overset{p1+p2}{\Longrightarrow} H$ via the parallel rule $p1 + p2$, such that we have

$$(G1, G2) \overset{I}{\rightarrowtail\!\!\Longrightarrow} G \overset{p1+p2}{\Longrightarrow} H = (G1, G2) \overset{(p1,p2)}{\Longrightarrow} (H1, H2) \overset{I}{\rightarrowtail\!\!\Longrightarrow} H$$

where $(G1, G2) \overset{(p1,p2)}{\Longrightarrow} (H1, H2)$ denotes the tupling of the separate transformations of $G1 \overset{p1}{\Longrightarrow} H1$ and $G2 \overset{p2}{\Longrightarrow} H2$. □

Proof Idea *The high-level replacement version of this theorem, given in [Pad96], can be applied to the category* **PARFUN**, *due to finite cocompleteness of the category* **PARFUN**, *see lemma A.2.* √

References

[AHS90] J. Adamek, H. Herrlich, and G. Strecker. *Abstract and Concrete Categories*. Series in Pure and Applied Mathematics. John Wiley and Sons, 1990.

[CLWW95] I. Claßen, M. Löwe, S. Waßerroth, and J. Wortmann. Algebraic Semantics for Object–oriented Modelling. Technical Report 95–10, Technische Universität Berlin, 1995.

[DE95] Jörg Desel and Javier Esparza. *Free Choice Petri Nets*. Cambridge University Press, 1995.

[DO96] J. Desel and A. Oberweis. Petrinetze in der angewandten Informatik. *Wirtschaftsinformatik*, 38(4):359–366, 1996.

[EHKP91] H. Ehrig, A. Habel, H.-J. Kreowski, and F. Parisi-Presicce. From graph grammars to High Level Replacement Systems. pages 269–291, 1991. Lecture Notes in Computer Science 532.

[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.

[EM90] H. Ehrig and B. Mahr. *Fundamentals of Algebraic Specification 2: Module Specifications and Constraints*, volume 21 of *EATCS Monographs on Theoretical Computer Science*. Springer, Berlin, 1990.

[EP97] H. Ehrig and J. Padberg. A Uniform Approach to Petri Nets. Springer, 1997. To appear.

[EPE96] C. Ermel, J. Padberg, and H. Ehrig. Requirements Engineering of a Medical Information System Using Rule-Based Refinement of Petri Nets. In *Proc. Integrated Design and Process Technology*, 1996. accepted.

[GGK96] G. Graw, V. Gruhn, and H. Krumm. Support of Cooperating and Distributed Business Processes. In *Proc. Int. Conf. on Parallel and Distributed Systems*, 1996. Tokyo.

[GH91] M. Gogolla and U. Hohenstein. Towards a Semantic View of an Extended Entity-Relationship Model. *ACM Transaction on Database Systems*, 16(2):369–416, 1991.

[GHMB93] D. Georgakopoulos, M.F. Hornick, F. Manola, M.L. Brodie, S. Heiler, F. Nayeri, and B. Hurwitz. *An Extended Transaction Environment for Workflows in Di stributed Object Computing. IEEE Data Engineering*, 16(2), September 1993.

[GL81] H.J. Genrich and K. Lautenbach. System modelling with high-level Petri nets. *Theoretical Computer Science*, 13:109–136, 1981.

[Gog94] M. Gogolla. *An Extended Entity-Relationship Model, Fundamentals and Pragmatics*, volume 767 of *LNCS*. Springer-Verlag, 1994.

[Gru91] V. Gruhn. *Validation and Verification of Software Process Models*. PhD thesis, Universität Dortmund, Abteilung Informatik, 1991.

[Gru96] V. Gruhn. Geschäftsprozeß-Management als Grundlage der Software-Entwicklung. *Informatik, Forschung und Entwicklung*, 11:94-101, 1996.

[GW95] V. Gruhn and S. Wolf. *Software Process Improvement by Business Process Orientation*. 1:49–56, August 1995. Pilot Issue.

[Het93] R. Hettler. Übersetzung von E/R-Modellen nach SPECTRUM. Technical Report TUM-I9333, Technische Universität München, 1993.

[HJS90] P. Huber, K. Jensen, and R.M. Shapiro. Hierarchies in Coloured Petri Nets. In G. Rozenberg, editor, *Advances in Petri nets, LNCS, vol. 483*, pages 313–341. Springer, 1990.

[Jen81] Kurt Jensen. Coloured Petri Nets and the Invariant Method. *Theoretical Computer Science*, 14:317–336, 1981.

[Jen92] Kurt Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use*, volume 1. Springer, 1992.

[KR96] E. Kindler and W. Reisig. Algebraic System Nets for Distributed Algorithms. Technical report, Humboldt University of Berlin, 1996.

[MR94] U. Montanari and F. Rossi. Contextual nets. To appear in Acta Informatica, 1994.

[NRT92] M. Nielsen, G. Rozenberg, and P.S. Thiagarajan. Elementary transition systems. *TCS*, 96:3–33, 1992.

[OSS94] A. Oberweis, G. Scherrer, and W. Stucky. INCOME/STAR: Methodology and Tools for the Development of Distributed Information Systems. *Information Systems*, 19(8):643–660, 1994.

[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.

[RBPEL91] J. Rumbaugh and M. Blaha and W. Premerlani and F. Eddy and W. Lorensen. *Object–oriented Modelling and Design.* Prentice–Hall International Editions. Prentice–Hall,, 1991.

[Rei85] W. Reisig. *Petri Nets*, volume 4 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, 1985.

[Rei91] W. Reisig. Petri nets and abstract data types. *Theoretical Computer Science*, **80**:1 – 34 (fundamental studies), April 1991.

[War94] B. Warboys. *Reflections on the Relationship Between BPR and Softwar e Process Modelling.* In P. Loucopoulos, editor, *Proceedings of the* 13^{th} *International Conference on the Entity-Relationship Approach*, pages 1–9, Manchester, UK, December 1994. Springer. Appeared as Lecture Notes in Computer Science no. 881.

[WRE95] H. Weber, W. Reisig, and H. Ehrig. Concept, Theoretical Foundation, and Validation of an Application Oriented Petri Net Technology. Proposal for a "Forschergruppe" to the German Research Council (DFG), 1995. (Accepted as a DFG-project from April 1996 to March 1999).