

Towards Partial Composition of Components: Formal Foundation for Component Verification

Julia Padberg

**Bericht-Nr. 2008-06
ISSN-Nummer: 1436-9915**

Towards Partial Composition of Components: Formal Foundation for Component Verification

Julia Padberg

Technische Universität Berlin
Fakultät IV – Informatik und Elektrotechnik
Franklinstr. 28/29
D-10587 Berlin

padberg@cs.tu-berlin.de

Abstract

The intention of this paper is to extend the generic component framework to partial composition. Basic Ideas of partial composition were introduced in [EM90] but we additionally want to allow component verification based on export-import implications. Import-Export implications relate sentences of the import stating what the component requires to sentences of the export stating what the component guarantees. The main result of this paper is the compatibility of import-export implications are compatible with the partial composition.

The second part illustrates how this abstract concept can be instantiated to Petri net systems.

1 Introduction

Components are self-contained entities that have a well-defined and mathematically precise syntax and semantics. In [EOB⁺02] a generic component framework for system modeling was introduced for a large class of semi-formal and formal modeling techniques. According to the transformation-based semantics [EOB⁺02] the notion of import-export implications characterize the Petri nets component with respect to its environment. In this report the concept of component based modeling and verification is extended to partial composition of components. A component is given by three specifications, the body specification, the import and the export interface. Formulating properties for components requires an adequate logic that allows expressing the desired properties. A component is then equipped with two temporal logic formulas that denote the import-export implication. The import assumptions describe in an abstract way the properties the underlying component needs to have to ensure the desired behavior. Then the export guarantees some property denoted by the export statement. Partial composition allows concluding the import-export implication where the providing component's import assumption implies only a part of the requiring component's export statement. In Fig. 1 to components and the result of their composition is illustrated, where the composition is achieved by gluing together the component bodies (i.e. the "clouds") along the interfaces.

According generic component framework [EOB⁺02] a component consists of a body, an import, and an export interface, and connections between import and body as well as export and body. We only require having suitable notions of embeddings and transformations (e.g. refinements) between specifications. This component technique is generic as it can be instantiated with different specification formalisms. Moreover, the connections can be considered generic as they also allow a great variety of instantiations. The basic idea for the generic component concept stems

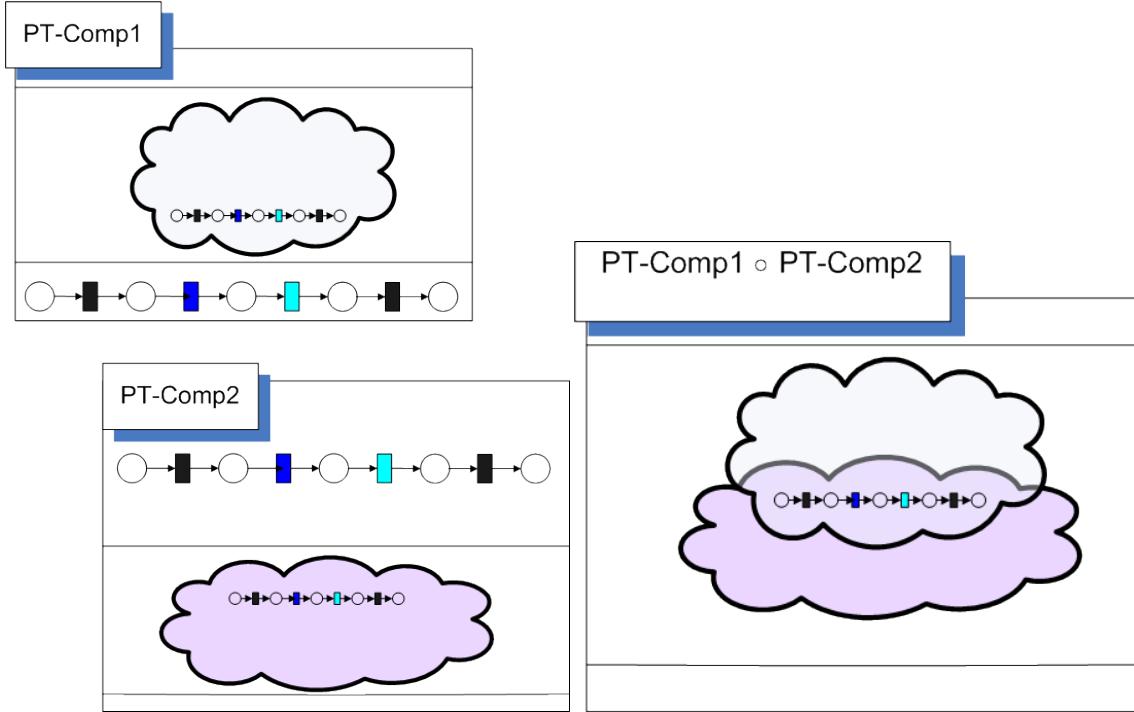


Figure 1: Illustration of composition

from data type specification, precisely the algebraic module specifications [EM90]. It was used for various related algebraic specification techniques as e.g. in [CBEO99, JO99]. The transfer to process description techniques was started in [Sim99] where modules for graph transformation systems and local action systems were investigated. In [Pad02] Petri net modules were introduced independently of the generic framework, but were shown to be compatible in [PE05]. In [EOB⁺02] algebraic high-level nets and in [Pad05] deterministic automata were demonstrated to be instantiations. In [PE07] we have extended the component concept with import-export implications of components that are formulas given in that temporal logic. The export statement given as part of the export interface is guaranteed independently of the component's environment provided the import requirement is met. This approach to component verification helps to guarantee specific properties that are formalized in terms of a temporal logic. So, we have an assume-guarantee approach to component-based verification that is independent of the underlying specification technique. This assume-guarantee approach is in model checking one approach to verification of components. There are various approaches, e.g. [IWY00, dAH01, GPB02] sharing the basic assumption that the required property can be achieved only in specific environments. In [GPB02] a framework for assume-guarantee paradigm is introduced that uses labeled transition systems to model the behavior of communicating components in a concurrent system. In [dAH01] the interfaces are modeled using input/output automata. The parallel composition of the interfaces is given and criteria for the compatibility are presented, but this approach merely concerns the interfaces. In [IWY00] certain properties, as deadlock freedom are checked based on assumptions that the component makes about the expected interaction behavior of other components. In [BM07] concurrent automata are introduced that describe the concurrent behavior of input and output ports in terms of their operations. Considering the automata as the components body and the input ans output ports as the import and export interfaces, respectively, maybe allows fitting this approach into the general framework presented in this paper. The area of controls for discrete event based systems needs an approach of modelling and structuring systems as well as the verification of the systems properties. In [PKA08] we propose to model and verify system properties of discrete event based systems using Petri nets and structure

them using multiple interfaces and partial composition in the component-based. We investigate the approach's feasibility for controlling a technical system and model the 'Compression Station' of a model plant for a packing process using Petri net components. The Petri net based sequence controller is modelled using the tool Netlab [Net07] which is a modelling, analysis and simulation environment that also supports the design and synthesis of discrete event - or hybrid systems under Matlab/Simulink. Netlab is a graphic PT net editor, loading and saving in PNML [OKA06].

This report is organized as follows: Next we review the generic component concept in detail and give the review of the hierarchical composition. Then in Section 3 we extend it by splitting the interfaces: so we have the possibility to express multiple interfaces. Next we transfer the verification results from [PEO07] to split imports. In Section 4 we instantiate the generic approach to Petri nets, and so have the basic modeling technique for the application in [PKA08], that is for modeling discrete event based systems and their verification. This application of the generic component based verification approach is most important for structuring the Petri nets hierarchically and to verifying their properties component-wise.

2 Component Verification for Generic Components

As the approaches in [EOB⁺02, EPB⁺04, EBK⁺05] this work employs generic specifications, embeddings and transformations to form components. Since not all classes of embeddings and transformations are suitable for this purpose we have to state some general requirements first. In the concrete specification technique the validity of these requirements needs to be proved when instantiating the generic concept.

2.1 General Assumptions of the Transformation based Approach

Our generic technique requires a defined class of specifications together with transformations and embeddings. The transformations define a class of refinements for the specifications, so they are used for the connection between export interface and the component body. Since there exist so many notions of refinement, even for a single specification technique, this assumption should not be further formalized at the abstract level. Nevertheless, it has to be spelled out for the instantiation of the concept. We now present the work concerning the generic concept of components in a categorical frame. In this framework a component consists of an import, an export and the body. The import states the prerequisites the component assumes. The body represents the internal functionality. The export gives an abstraction of the body that can be used by the environment. In [PE05] we present a categorical formalization of the concepts of the generic framework using specific kinds of pushout properties.

Definition 2.1 (Generic framework \mathcal{T} for components)

A generic framework for components $\mathcal{T} = (\mathbf{Cat}, \mathcal{I}, \mathcal{E})$ consists of an arbitrary category and two classes of morphisms \mathcal{I} , called import morphisms and \mathcal{E} , called export morphisms such that the following *extension conditions* hold:

1. Existence of Extension Diagrams:

Extension diagrams are commutative squares as (1) that are pushouts.

2. \mathcal{E} - \mathcal{I} -Pushout Condition:

Given the morphisms $A \xrightarrow{e} B$ with $e \in \mathcal{E}$ and $A \xrightarrow{i} C$ with $i \in \mathcal{I}$, then there exists the pushout D in **Cat** with morphisms $B \xrightarrow{i'} D$ and $C \xrightarrow{e'} D$ as depicted adjacently.

$$\begin{array}{ccc} A & \xrightarrow{e} & B \\ i \downarrow & (1) & \downarrow i' \\ C & \xrightarrow{e'} & D \end{array}$$

3. \mathcal{E} and \mathcal{I} are stable under pushouts:

Given a \mathcal{E} - \mathcal{I} -pushout as (1) above, then we have $i' \in \mathcal{I}$ and $e' \in \mathcal{E}$ as well. \diamond

Note that this definitions differs slightly from [PE05], since here we made explicit the existence of extension diagrams as specific pushouts.

Definition 2.2 (Import-export implication for components)

Given a component $Comp = (IMP, EXP, BOD, imp, exp)$ then an import-export implication $\rho \Rightarrow \gamma$ consists of $\rho \in Sen(IMP)$ and $\gamma \in Sen(EXP)$. \diamond

The import-export implication provides information on the component's body at its interfaces. This information concerns the assumptions and guarantees of a component in an arbitrary environment. So, satisfaction of the import-export implications is formulated with respect to an arbitrary environment, formalized by an arbitrary transformation of the import interface. Then we require that if this environment satisfies the translated import assumption, then the corresponding export will satisfy the translated export statement.

Definition 2.3 (Satisfaction of an import-export implication)

Given a component $Comp = (IMP, EXP, BOD, imp, exp)$ then

the import-export implication $\rho \Rightarrow \gamma$ with $\rho \in Sen(IMP)$ and

$\gamma \in Sen(EXP)$ is satisfied if we have

$SPEC \models \mathfrak{T}_{trafo}(\rho) \Rightarrow SPEC' \models \mathfrak{T}_{trafo' \circ exp}(\gamma)$

for all extension diagrams:

$$\begin{array}{ccccc} & & EXP & & \\ & & \exp \parallel & & \\ & & \downarrow & & \\ IMP & \xrightarrow{imp} & BOD & & \\ trafo \downarrow & & \downarrow trafo' & & \\ SPEC & \xrightarrow{imp'} & SPEC' & & \end{array}$$

\diamond

A component with guarantees is a component that ensures the export statement for any possible environment provided the import assumptions are met.

Definition 2.4 (Component with guarantees)

A component with guarantees $Comp = (IMP, EXP, BOD, imp, exp, \rho, \gamma)$ consists of a component $(IMP, EXP, BOD, imp, exp)$ together with the import-export implication $\rho \Rightarrow \gamma$ that has to be satisfied. \diamond

Then hierarchical composition allows the propagation of the export statements, provided the export statement of the imported component implies the import requirement of the importing component. This is defined by the connecting condition.

Definition 2.5 (Connecting condition)

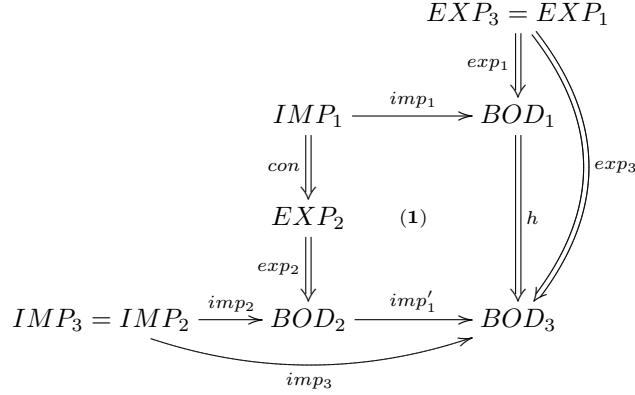
Given components $Comp_i = (IMP_i, EXP_i, BOD_i, imp_i, exp_i, \gamma_i, \rho_i)$ for $i \in \{1, 2\}$ and a connection transformation $con : IMP_1 \Rightarrow EXP_2$ then the connecting condition is satisfied if we have for all transformations $trafo : EXP_2 \Rightarrow SPEC$:

$$SPEC \models \mathfrak{T}_{trafo}(\gamma_2) \Rightarrow SPEC \models \mathfrak{T}_{trafo \circ con}(\rho_1)$$

\diamond

Definition 2.6 (Hierarchical Composition)

Given components $Comp_i = (IMP_i, EXP_i, BOD_i, imp_i, exp_i, \gamma_i, \rho_i)$ for $i \in \{1, 2\}$ and a connection transformation $con : IMP_1 \Rightarrow EXP_2$ then the hierarchical composition $Comp_3$ of $Comp_1$ and $Comp_2$ via $con : IMP_1 \Rightarrow EXP_2$ is defined by $Comp_3 := Comp_1 \circ_{con} Comp_2 = (IMP_3, EXP_3, BOD_3, imp_3, exp_3, \gamma_1, \rho_2)$ with $imp_3 := imp_1 \circ imp_2$ and $exp_3 := h \circ exp_1$ as depicted below where (1) is an extension diagram :



◇

In order to have a compositional approach to component verification we now need to ensure that the hierarchical composition preserves the components guarantees in a suitable way.

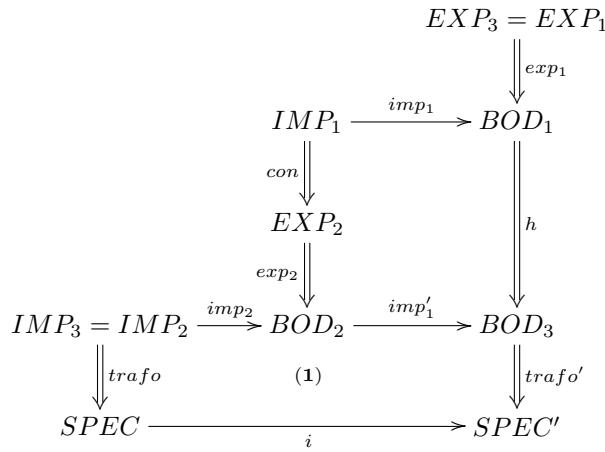
Fact 2.7 (Hierarchical composition propagates guarantees)

Given components $Comp_1$ and $Comp_2$ with guarantees and a connection $con : IMP_1 \Rightarrow EXP_2$ satisfying the connecting condition, then the result of the hierarchical composition $Comp_3 = Comp_1 \circ_{con} Comp_2$ is again a component with guarantees. ◇

So, we have to show that the hierarchical composition $Comp_3 = Comp_1 \circ_{con} Comp_2$ satisfies the import-export implication $\rho_2 \Rightarrow \gamma_1$.

Proof:

We need to show that $SPEC \models \Sigma_{trafo}(\rho_2) \Rightarrow SPEC' \models \Sigma_{trafo' \circ exp_3}(\gamma_1)$ for any extension (1) in the diagram below:



Due to pushout decomposition we have the three extension diagrams (2), (3) and (4) with $i = i_1 \circ i_2$:

$$\begin{array}{ccccc}
& & EXP_3 = EXP_1 & & \\
& & \downarrow exp_1 & & \\
& & IMP_1 \xrightarrow{imp_1} BOD_1 & & \\
& & \downarrow con & & \downarrow h \\
& & EXP_2 & (2) & \\
& & \downarrow exp_2 & & \downarrow \\
IMP_3 = IMP_2 \xrightarrow{imp_2} BOD_2 \xrightarrow{imp'_1} BOD_3 & & \downarrow trafo'' \quad (4) & & \downarrow trafo' \\
\downarrow trafo \quad (3) & & \downarrow trafo'' \quad (4) & & \downarrow trafo' \\
SPEC \xrightarrow{i_2} SPEC'' \xrightarrow{i_1} SPEC' & & \searrow i & &
\end{array}$$

So, we have:

$$\begin{aligned}
SPEC \models \mathfrak{T}_{trafo}(\rho_2) \Rightarrow SPEC'' \models \mathfrak{T}_{trafo'' \circ exp_2}(\gamma_2) & \quad \text{as } Comp_2 \text{ has guarantees,} \\
SPEC'' \models \mathfrak{T}_{trafo'' \circ exp_2}(\gamma_2) \Rightarrow SPEC'' \models \mathfrak{T}_{trafo'' \circ exp_2 \circ con}(\rho_1) & \quad \text{due to the connecting condition, and} \\
SPEC'' \models \mathfrak{T}_{trafo'' \circ exp_2 \circ con}(\rho_1) \Rightarrow SPEC' \models \mathfrak{T}_{trafo' \circ h \circ exp_1}(\gamma_1) & \quad \text{as } Comp_1 \text{ has guarantees and due to composition of pushouts}
\end{aligned}$$

So, we directly conclude:

$$SPEC \models \mathfrak{T}_{trafo}(\rho_2) \Rightarrow SPEC' \models \mathfrak{T}_{trafo' \circ h \circ exp_1}(\gamma_1)$$

✓

3 Components and Partial Composition

Accordingly, we have to require for a component that the import and export connection are of the right class of morphisms.

Definition 3.1 (Component)

A component specification $Comp = (IMP, EXP, BOD, imp, exp)$ consists of a

body specification BOD , an import specification IMP with an embedding $IMP \xrightarrow{imp} BOD$ and an export specification EXP with a transformation $EXP \xrightarrow{exp} BOD$.

$$\begin{array}{ccc}
& EXP & \\
& \downarrow exp & \diamond \\
IMP \xrightarrow{imp} BOD & &
\end{array}$$

Here, we give examples of components and their partial composition in terms of Petri net components as defined in Section 4. Figure 2 gives the component $COMP1$ consisting of export $EXP1$, import $IMP1$ and body $BOD1$.

Then partial composition allows the splitting of the import into two (or more) import parts, provided the imported component's import can be glued adequately to those import parts that are not used. for the gluing of the import interface we need a specific morphisms, so called gluing morphisms. Fig. 3 illustrates the partial connection based on a pushout in the category of PT net systems.

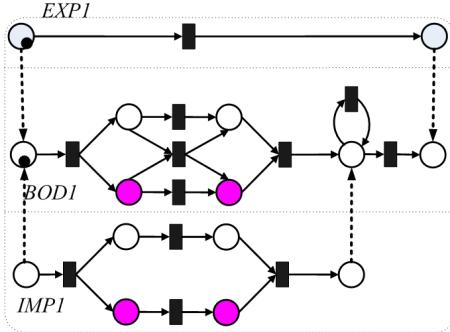


Figure 2: Component $Compl_1$

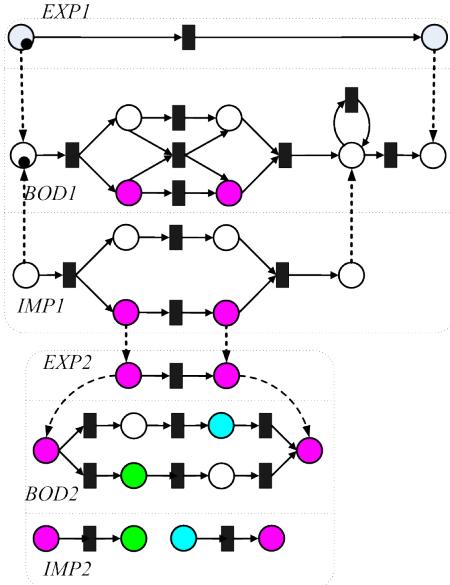


Figure 3: Partial Connection of two Components

Definition 3.2 (Generic framework \mathcal{T} for components with split imports)

A generic framework $\mathcal{T}_P = (\mathbf{Cat}, \mathcal{I}, \mathcal{E}, \mathcal{G})$ for components with split imports is a generic framework $(\mathbf{Cat}, \mathcal{I}, \mathcal{E})$ and additionally consists of a class of gluing morphisms $\mathcal{G} \subseteq \mathcal{I}$ – denoted by $-->$ such that the following *conditions* hold:

1. For $g \in \mathcal{G}$ and $f \in \mathcal{I}$ we have $(f \circ g) \in \mathcal{G}$ as well, and
2. the *induced import condition* holds: $imp_3 \in \mathcal{I}$ in the diagram of Def. 3.5.

◇

Definition 3.3 (Split Import)

Given the component $Comp = (IMP, EXP, BOD, imp, exp)$ then a split import is given by the pushout $IMP = IMP_1 +_I IMP_2$, where $I --> IMP_{1.1}$ and $I --> IMP_{1.2}$ are in \mathcal{G} . We call this pushout a \mathcal{G} -pushout. ◇

Definition 3.4 (Partial Composition Connection)

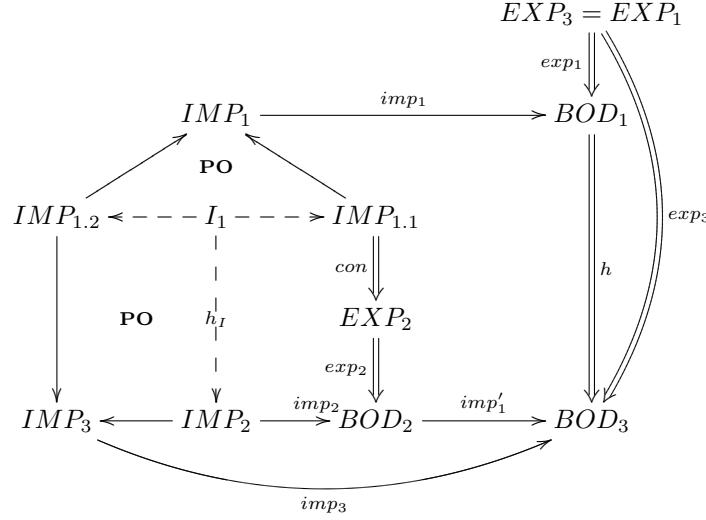
Given components $Comp_i = (IMP_i, EXP_i, BOD_i, imp_i, exp_i)$ for $i \in \{1, 2\}$ with a split import given by the \mathcal{G} -pushout $IMP_i = IMP_{i.1} +_{I_i} IMP_{i.2}$,

then a connection is given by a transformation $con : IMP_{1.1} \Rightarrow EXP_2$ and \mathcal{G} -morphism $h_I : I_1 \rightarrow IMP_2$, so that $I_1 \dashrightarrow IMP_{1.1} \xrightarrow{con \circ exp_2} BOD_2 = I_1 \dashrightarrow IMP_2 \rightarrow BOD_2$ \diamond

Fact 3.5 (Partial Composition)

Given components $Comp_i = (IMP_i, EXP_i, BOD_i, imp_i, exp_i)$ for $i \in \{1, 2\}$ with the split import given by the \mathcal{G} -pushout $IMP_1 = IMP_{1.1} +_{I_1} IMP_{1.2}$ and a connection

– given by a transformation $con : IMP_{1.1} \Rightarrow EXP_2$ and $h_I \in \mathcal{G} : I_1 \rightarrow IMP_2$ – then the partial composition $Comp_3$ of $Comp_1$ and $Comp_2$ via (con, h_I) given by $Comp_3 := Comp_1 \circ_{con, h_I} Comp_2 = (IMP_3, EXP_3, BOD_3, imp_3, exp_3)$ is a well-defined component as depicted below with a split import given by $IMP_3 = IMP_{1.2} +_{I_1} IMP_2$:



$imp_3 \in \mathcal{I}$ due to condition 2 of Definition 3.2. \diamond

Proof:

Due to the fact that $Comp_3 = (IMP_3, EXP_3, BOD_3, imp_3, exp_3)$ is a component with $exp_3 \in \mathcal{E}$ and $imp_3 \in \mathcal{I}$. \checkmark

Remark: Disjointly split imports

The disjointly split import is merely a special case of the split import, provided the underlying category has initial objects and the induced morphisms are in \mathcal{G} . \triangle

Figure 4 illustrates the construction of the new import interface IMP_3 of the component $COMP_3 = COMP_1 \circ COMP_2$ and the component itself in Figure 3.

3.1 Import-Export Implications

Components are self-contained units with a well-defined syntax and semantics. In [EOB⁺02] semantics of components are defined by considering each possible environment expressed by each possible transformation of the component's import. According to the transformation-based semantics the notion of import-export implications characterize the component with respect to its environment. Based on an adequate logic calculus that allows the formulation of formulas and their translation along transformations, import-export implications can be defined for components. To define a logic over a specification we need to relate the vocabulary of the logic to the specification $SPEC$, so we need some signature Σ for $SPEC$. Then $SPEC \in \Sigma$ the set of all specifications with signature Σ .

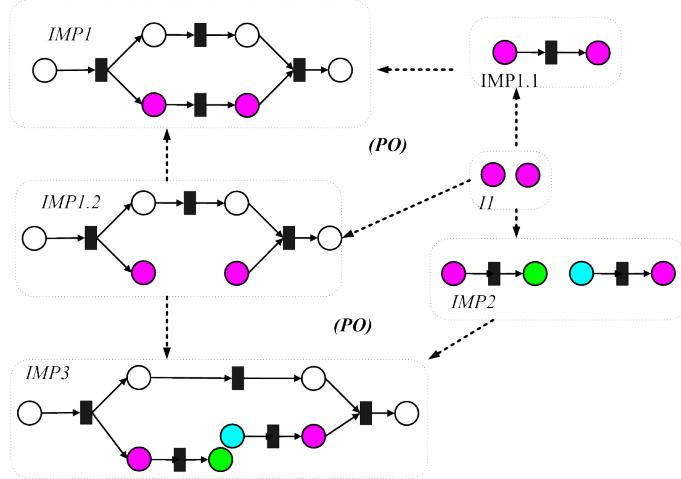


Figure 4: Construction of $IMP3$

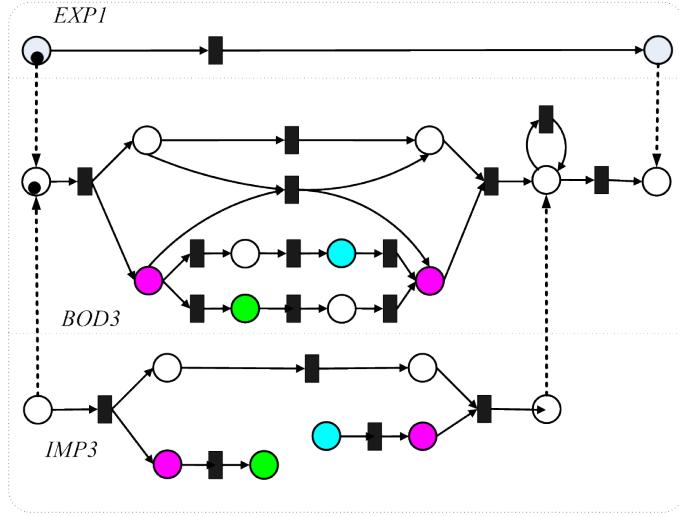


Figure 5: Component $COMP3$

Definition 3.6 (Underlying logic)

The underlying logic $(Sen(\Sigma), \models)$ over the signature Σ consists of the set of formulas over that signature $Sen(\Sigma)$ and a relation $\models_{\Sigma} \subseteq \Sigma \times Sen(\Sigma)$ where Σ denotes the set of all specifications with signature Σ . \diamond

Definition 3.7 (Translation of the underlying logic)

Given the underlying logic $(Sen(\Sigma), \models)$ then for each transformation $trafo : SPEC_1 \rightarrow SPEC_2$ there has to be a translation of sentences $\mathfrak{T}_{trafo} : Sen(\Sigma_1) \rightarrow Sen(\Sigma_2)$ with $SPEC_i \in \Sigma_i$ for $1 \leq i \leq 2$.

The translation has to be compatible with the morphism composition, i.e. for transformations $trafo_i : SPEC_i \rightarrow SPEC_{i+1}$ with $i \leq i \leq 2$ there is the translation $\mathfrak{T}_{trafo_1 \circ trafo_2} = \mathfrak{T}_{trafo_1} \circ \mathfrak{T}_{trafo_2} : Sen(\Sigma_1) \rightarrow Sen(\Sigma_3)$ for $SPEC_i \in \Sigma_i$.

The translation along an identity has to yield an identical translation, i.e. $\mathfrak{T}_{id} = \mathfrak{I}\mathfrak{D}$. \diamond

Note that $SPEC \models \varphi$ then $SPEC' \models \mathfrak{T}_{trafo}(\varphi)$ is not demanded as it is too strong for most process specification. E.g. liveness considered as a temporal logic formula over some process specification is usually not preserved by morphisms.

3.2 Component with guarantees for split imports

Definition 3.8 (Import-export implication for components with split import)

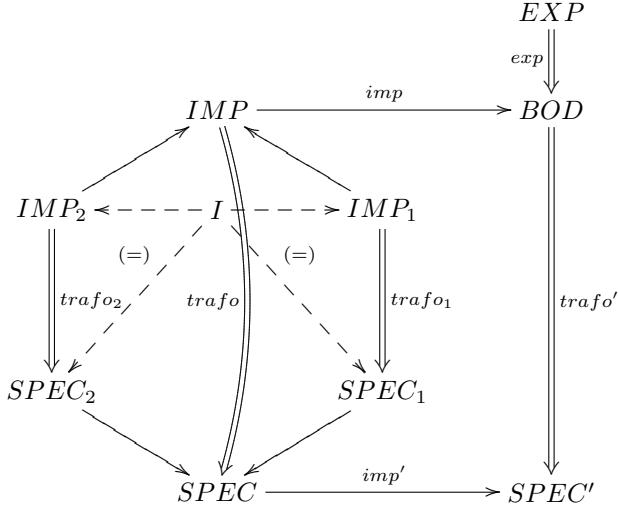
Given a component $Comp = (IMP, EXP, BOD, imp, exp, \rho_1 \wedge \rho_2, \gamma)$ with a split import given by the \mathcal{G} -pushout $IMP = IMP_1 +_I IMP_2$ then the split import-export implication with $\rho_1 \wedge \rho_2 \Rightarrow \gamma$ has to consist of $\rho_1 \in Sen(IMP_1)$, $\rho_2 \in Sen(IMP_2)$ and $\gamma \in Sen(EXP)$. \diamond

Definition 3.9 (Satisfaction of a split import-export implication)

Given a component $Comp = (IMP, EXP, BOD, imp, exp, \rho_1 \wedge \rho_2, \gamma)$ with a split import given by the G -pushout $IMP = IMP_1 +_I IMP_2$, then split import-export implication $\rho_1 \wedge \rho_2 \Rightarrow \gamma$ consisting of $\rho_1 \in Sen(IMP_1)$, $\rho_2 \in Sen(IMP_2)$ and $\gamma \in Sen(EXP)$ is satisfied if

$$SPEC_1 \models \mathfrak{T}_{trafo_1}(\rho_1) \wedge SPEC_2 \models \mathfrak{T}_{trafo_2}(\rho_2) \Rightarrow SPEC' \models \mathfrak{T}_{trafo' \circ exp}(\gamma)$$

for all extension diagrams:



where we have the \mathcal{G} -pushout $SPEC = SPEC_1 +_1 SPEC_2$, the induced \mathcal{E} -morphism $IMP \Longrightarrow SPEC$ and the extension pushout $SPEC' = SPEC +_{IMP} BOD$. \diamond

A component with guarantees is a component that ensures the export statement for any possible environment provided the import assumptions are met.

Definition 3.10 (Split Component with guarantees)

Shortly, we denote a component with split import and split import-export implication by $Comp = (IMP_1 +_I IMP_2, EXP, BOD, imp, exp, \rho_1 \wedge \rho_2, \gamma)$ and assume the \mathcal{G} -pushout $IMP = IMP_1 +_I IMP_2$ and that $\rho_1 \in Sen(IMP_1)$ and $\rho_2 \in Sen(IMP_2)$. Moreover, we say that $Comp$ is a split component with guarantees if it satisfies the split import-export implication. \diamond

Definition 3.11 (Connecting condition)

Given a component $Comp_1 = (MP_{1.1} +_{I_1} IMP_{1.2}, EXP_1, BOD_2, imp_1, exp_2, \rho_{1.1} \wedge \rho_{1.2}, \gamma_1)$ and a component $Comp_2 = (IMP_2, EXP_2, BOD_2, imp_2, exp_2, \rho_2, \gamma_2)$ and a connection transformation $con : IMP_{1.1} \Longrightarrow EXP_2$ and $h_I : I_1 \rightarrow IMP_2$ then the connecting condition is satisfied if we have for all transformations $trafo : EXP_2 \Longrightarrow SPEC$:

$$SPEC \models \mathfrak{T}_{trafo}(\gamma_2) \Rightarrow SPEC \models \mathfrak{T}_{trafo \circ con}(\rho_1)$$

\diamond

In order to have a compositional approach to component verification we now need to ensure that the partial composition preserves the components guarantees in a suitable way.

Fact 3.12 (Partial composition propagates guarantees)

Given a split component $Comp_1 = (IMP_{1.1} +_{I_1} IMP_{1.2}, EXP_1, BOD_2, imp_1, exp_2, \rho_{1.1} \wedge \rho_{1.2}, \gamma_1)$ with guarantees and a component $Comp_2 = (IMP_2, EXP_2, BOD_2, imp_2, exp_2, \rho_2, \gamma_2)$ with guarantees and a connection

- given by a transformation $con : IMP_{1.1} \Rightarrow EXP_2$ and a \mathcal{G} -morphism $h_I : I_1 \rightarrow IMP_2$ – satisfying the connecting condition

then the partial composition $Comp_3$ of $Comp_1$ and $Comp_2$ is defined by

$$Comp_3 := Comp_1 \circ_{con, h_I} Comp_2 = (IMP_3, EXP_3, BOD_3, imp_3, exp_3, \rho_2 \wedge \rho_{1.2}, \gamma_1)$$

where

IMP_3 is given by the \mathcal{G} -pushout $IMP_3 = IMP_{1.2} +_{I_1} IMP_2$,

$BOD_3 = BOD_1 +_{IMP_{1.1}} BOD_2$ is pushout,

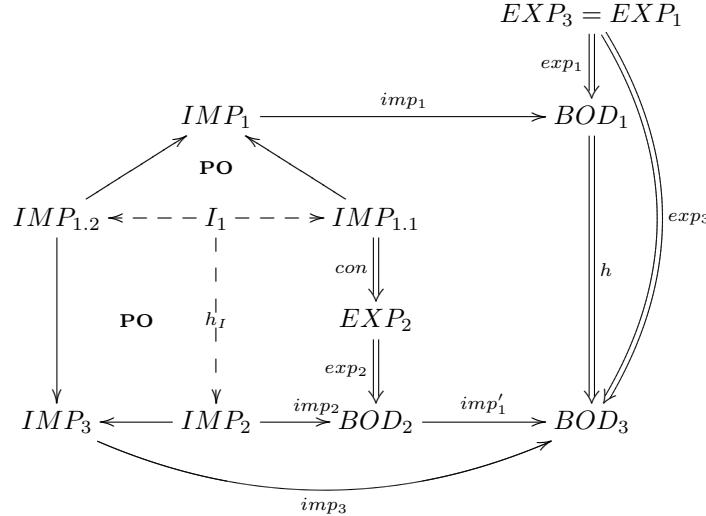
imp_3 is induced pushout morphism $imp_3 : IMP_3 \rightarrow BOD_3$ and

$\rho_{1.2} \in Sen(IMP_{1.2})$, $\rho_2 \in Sen(IMP_2)$ and $\gamma_1 \in Sen(EXP_1)$

is again a split component with guarantees. \diamond

Proof:

We have the following construction:



We need to show that for $Comp_3 = (IMP_3, EXP_3, BOD_3, imp_3, exp_3, \rho_2 \wedge \rho_{1.2}, \gamma_1)$ the import-export implication $\rho_2 \wedge \rho_{1.2} \Rightarrow \gamma_1$ consisting of $\rho_{1.2} \in Sen(IMP_{1.2})$, $\rho_2 \in Sen(IMP_2)$ and $\gamma_1 \in Sen(EXP_1)$ is satisfied:

$$SPEC_{1.2} \models \mathfrak{T}_{trafo_1}(\rho_{1.2}) \wedge SPEC_2 \models \mathfrak{T}_{trafo_2}(\rho_2) \Rightarrow SPEC'_3 \models \mathfrak{T}_{trafo_3 \circ exp_3}(\gamma_1)$$

We have by the assumptions the following situation and the following pushouts:

$SPEC_{1.2} \models \rho_{1.2}$ and $SPEC_2 \models \rho_2$ by assumption

PO1: $IMP_1 = IMP_{1.1} +_{I_1} IMP_{1.2}$

PO2: $IMP_3 = IMP_2 +_{I_1} IMP_{1.2}$

PO3: $BOD_3 = BOD_1 +_{IMP_{1.1}} BOD_2$

PO4: $SPEC'_3 = SPEC_3 +_{IMP_3} BOD_3$

We obtain $IMP_1 \rightarrow \widehat{SPEC}_1$ by **PO1** and **PO6**. Due to uniqueness of induced pushout morphism, for we have

- $IMP_{1.1} \rightarrow IMP_1 \rightarrow BOD_1 \Rightarrow SPEC'_3 = IMP_{1.1} \Rightarrow SPEC'_2 \rightarrow SPEC'_3$
- $IMP_{1.1} \rightarrow IMP_1 \Rightarrow \widehat{SPEC}_1 \rightarrow SPEC'_3 = IMP_{1.1} \Rightarrow SPEC'_2 \rightarrow SPEC'_3$
- $IMP_{1.2} \rightarrow IMP_1 \rightarrow BOD_1 \Rightarrow SPEC'_3 = IMP_{1.2} \Rightarrow SPEC_{1.2} \rightarrow SPEC'_3$
- $IMP_{1.2} \rightarrow IMP_1 \Rightarrow \widehat{SPEC}_1 \rightarrow SPEC'_3 = IMP_{1.2} \Rightarrow SPEC_{1.2} \rightarrow SPEC'_3$

there is the communicating square (8)

$$IMP_1 \rightarrow BOD_1 \Rightarrow SPEC'_3 = IMP_1 \Rightarrow \widehat{SPEC}_1 \rightarrow SPEC'_3.$$

We show (8) to be pushout, namely **PO8**.

Given X with $IMP_1 \rightarrow BOD_1 \Rightarrow X = IMP_1 \Rightarrow \widehat{SPEC}_1 \rightarrow X$ as depicted below we obtain $BOD_3 \Rightarrow X$ due to **PO3**.

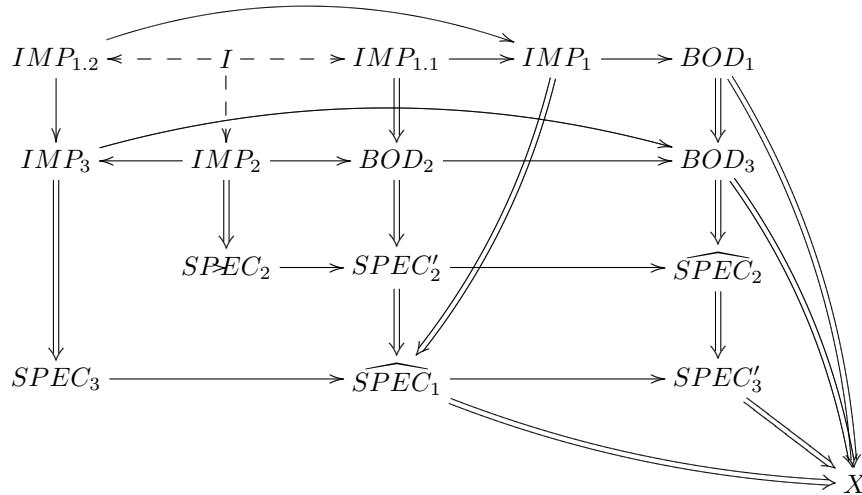
Due to uniqueness of the pushout morphisms for

- $IMP_{1.2} \rightarrow IMP_3 \rightarrow BOD_3 \Rightarrow X = IMP_{1.2} \rightarrow IMP_1 \rightarrow BOD_1 \Rightarrow X$
- $IMP_{1.2} \rightarrow IMP_3 \rightarrow SPEC_3 \rightarrow \widehat{SPEC}_1 \Rightarrow X = IMP_{1.2} \rightarrow IMP_1 \rightarrow BOD_1 \Rightarrow X$
- $IMP_2 \rightarrow IMP_3 \rightarrow BOD_3 \Rightarrow X = IMP_2 \Rightarrow SPEC_2 \rightarrow SPEC_3 \rightarrow \widehat{SPEC}_1 \Rightarrow X$
- $IMP_2 \rightarrow IMP_3 \rightarrow SPEC_3 \rightarrow \widehat{SPEC}_1 \Rightarrow X$
 $= IMP_2 \Rightarrow SPEC_2 \rightarrow SPEC_3 \rightarrow \widehat{SPEC}_1 \Rightarrow X$

we have $IMP_3 \rightarrow BOD_3 \Rightarrow X = IMP_3 \Rightarrow SPEC_3 \rightarrow \widehat{SPEC}_1 \Rightarrow X$.

And so **PO2** induces $SPEC'_3 \Rightarrow X$ that uniquely commutes:

$$BOD_1 \Rightarrow SPEC'_3 \Rightarrow X = BOD_1 \Rightarrow X \text{ and } \widehat{SPEC}_1 \rightarrow SPEC'_3 \Rightarrow X = \widehat{SPEC}_1 \Rightarrow X$$



So, **PO8** is pushout and hence we have:

$$SPEC_{1.2} \models \rho_{1.2} \wedge SPEC'_2 \models \rho_{1.1} \Rightarrow SPEC'_3 \models \gamma_1$$

✓

This means $Comp_3$ satisfies the import-export implication.

4 Instantiation to PT Net Systems

First we give a short intuition of the underlying formalism. We use the algebraic notion of Petri nets as introduced in [MM90]. Hence, a place/transition (PT) net is given by the set of transitions and the set of places and the pre and post domain function; $N = (T \xrightarrow[\text{post}]{\text{pre}} P^\oplus)$ where P^\oplus is the free commutative monoid over P_- or the set of finite multisets over P . So, an element $w \in P^\oplus$ can be presented as a linear sum $w = \sum_{p \in P} \lambda_p p$ and we can extend the usual operations and relations as \oplus, \ominus, \leq , and so on. The initial marking (and markings in general) can be understood both as a linear sum, i.e. $\hat{m} \in P^\oplus$ as well as a finitely based mapping, i.e. $\hat{m} : P \rightarrow \mathbb{N}$.

We use much simpler morphisms than in [Pad02] that do not preserve any specific properties as safety or liveness. The import morphism *imp* is a *plain, injective morphism* and describes where the resources of the import are used in the body. The export morphism *exp* is a *t-partial, injective morphism*. So, we have a very loose interpretation of refinement: those transition that are not mapped represent some not explicitly specified subnet of the target net.

Definition 4.1 (PT net system and morphisms classes)

A place/transition net system is given by $PS = (T \xrightarrow[\text{post}]{\text{pre}} P^\oplus, \hat{m})$

- where P^\oplus is the free commutative monoid over P
- and $\hat{m} \in P^\oplus$ is the initial marking.
(it may as well be given as a finitely based mapping, i.e. $\hat{m} : P \rightarrow \mathbb{N}$)

We have the subsequent morphism classes:

- A t-partial morphisms $h : PS_1 \rightarrow PS_2$ is a mapping where $h_P : P_1 \rightarrow P_2$ is a total function and $h_T : T_1 \rightarrow T_2$ is a partial function such that h is arc preserving; for all $t \in \text{dom}(f_T)$ we have: $h_P^\oplus \circ \text{pre}_1 = \text{pre}_2 \circ h_T(t)$ and $h_P^\oplus \circ \text{post}_1 = \text{post}_2 \circ f_T(t)$.
- Morphisms are plain if $h_T : T_1 \rightarrow T_2$ is a total function as well. The class of injective plain morphisms is denoted by \mathcal{I} .
- Morphisms are marking strict if $\widehat{m_1}(p) = \widehat{m_2}(h(p))$ for all $p \in P_1$. The class of marking strict t-partial, injective morphisms is denoted by \mathcal{E} .
- PT net systems and t-partial morphism comprise the category $\mathbf{PS}_{\mathbf{tp}}$.
- a gluing morphisms is given by $g : PS_1 \rightarrow PS_2$ is an injective mapping of places only with $h_P : P_1 \rightarrow P_2$ a total function and $T_1 = \emptyset$ and accordingly $g_T : \emptyset \rightarrow T_2$ the empty function. These morphisms constitute the class \mathcal{G} .

◇

Based on the category $\mathbf{PS}_{\mathbf{tp}}$ and the above morphisms classes form an instance of the generic framework for components with split imports and accordingly provide the same results (see Results 4.10).

Fact 4.2 (Generic framework \mathcal{PT}_P for PN-components with split imports)

The category $\mathbf{PS}_{\mathbf{tp}}$ with the morphism classes \mathcal{E}, \mathcal{I} and \mathcal{G} as defined in Definition 4.1 is a generic framework $\mathcal{T}_P = (\mathbf{Cat}, \mathcal{I}, \mathcal{E}, \mathcal{G})$ for components with split imports. ◇

Proof:

1. Existence of Extension Diagrams (see Fact 4.4).

2. \mathcal{E} - \mathcal{I} -Pushout Condition (see [Pad06])
3. \mathcal{E} and \mathcal{I} are stable under pushouts (see [Pad06])
4. For $g \in \mathcal{G}$ we have $(f \circ g) \in \mathcal{G}$ as well due to the definition of \mathcal{G} morphisms.
5. The induced import condition holds in **Set**, see Fact 4.6.

✓

Fact 4.3 (Pushouts in the category $\mathbf{PS}_{\mathbf{tp}}$ [Pad06])

Given PT systems PS_i for $0 \leq i \leq 2$ and morphisms $PS_1 \xleftarrow{h_1} PS_0 \xrightarrow{g_1} PS_2$ in the category $\mathbf{PS}_{\mathbf{tp}}$ then here is a pushout $PS_1 \xrightarrow{h_2} PS_3 \xleftarrow{g_2} PS_2$ that can be constructed component-wise for places and transitions and with an arbitrary initial marking for PS_3 . \diamond

Fact 4.4 (Extension diagrams of \mathcal{I} - and \mathcal{E} -morphisms [Pad06])

Given PT nets $N_i = (P_i, T_i, pre_i, post_i)$ and PT systems $PS_i = (N_i, \widehat{m_i})$ and the morphisms $PS_1 \xleftarrow{h_1} PS_0 \xrightarrow{g_1} PS_2$ where $h_1 \in \mathcal{I}$ and $g_1 \in \mathcal{E}$ then there is a pushout $PS_1 \xrightarrow{h_2} PS_3 \xleftarrow{g_2} PS_2$ with

$$\widehat{m}_3(p) = \begin{cases} \widehat{m}_2(p_2) & ; g_2(p_2) = p \notin h_2(P_1) \\ \widehat{m}_1(p_1) & ; h_2(p_1) = p \end{cases}$$

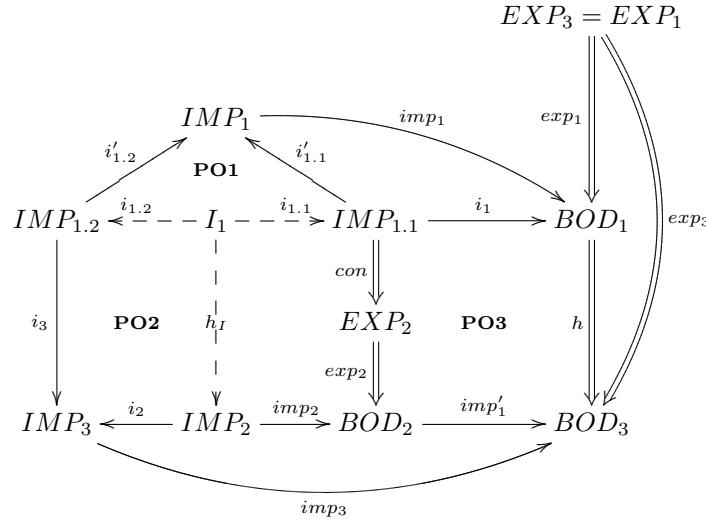
that is an extension diagram with $g_2 \in \mathcal{I}$ and $h_2 \in \mathcal{E}$. \diamond

Fact 4.5 (Induced import condition in Set)

◊

Proof:

We have in **Set**, where \mathcal{I} is the class of injective and \mathcal{E} the class of total functions the following diagram in **Set**:



We show indirectly that imp_3 is injective, using the following cases:

Let be given $x \neq y \in Imp_3$ and $imp_3(x) = imp_3(y)$ with

1. $x = i_2(x')$ and $y = i_2(y')$ for $x', y' \in IMP_2$

then $x' \neq y'$ and – as imp'_1 and imp_2 are injective – we have $imp_3(x) = imp_3 \circ i_2(x') = imp'_1 \circ imp_2(x') \neq imp'_1 \circ imp_2(y') = imp_3 \circ i_2(y') = imp_3(y)$ which contradicts the assumption \nexists

2. $x = i_3(x')$ and $y = i_3(y')$ for $x', y' \in IMP_{1.2}$ and $\neg \exists z \in IMP_2 : i_2(z) = x \vee i_2(z) = y$
then $\neg \exists z \in I_1 : i_2(z) = x \vee i_2(z) = y$ due to pushout **PO2** and $x' \neq y'$. Hence, there are $imp_1 \circ i'_{1.2}(x')$, $imp_1 \circ i'_{1.2}(y') \in BOD_1$ with $imp_1 \circ i'_{1.2}(x'), imp_1 \circ i'_{1.2}(y') \notin i_1(IMP_{1.1})$ due to pushout **PO1**.
Due to the construction of semi-injective pushouts $h : BOD_{1|_{i_1(IMP_{1.1})}} \rightarrow BOD_3$ is injective, and hence we have
 $imp_3(x) = imp_3 \circ i_3(x') = h \circ imp_1 \circ i'_{1.2}(x') \neq h \circ imp_1 \circ i'_{1.2}(y') = imp_3 \circ i_3(y') = imp_3(y)$
which contradicts the assumption \nexists

3. $x = i_3(x')$ for $x' \in IMP_{1.2}$ and $\neg \exists z \in IMP_2 : i_2(z) = x$ and $y = i_2(y')$ for $y' \in IMP_2$
Hence $imp_1 \circ i'_{1.2}(x') \in BOD_1 \setminus (i_1(IMP_{1.1}))$ and $imp_2(y') \in BOD_2$. Since $BOD_3 = BOD_2 \uplus BOD_1 \setminus (i_1(IMP_{1.1}))$ we have $hcirc imp_1 \circ i'_{1.2}(x') \neq imp'_1 \circ imp_2(y')$, and so
 $imp_3(x) = imp_3 \circ i_3(x') = h \circ imp_1 \circ i'_{1.2}(x') \neq imp'_1 \circ imp_2(y') = imp_3 \circ i_2(y') = imp_3(y)$
which contradicts the assumption \nexists

✓

Fact 4.6 (Induced import condition in PS_{tp})

◇

Proof:

Since we have the induced import condition in **Set** and PT net systems and the morphisms are given component wise, we can conclude that both $imp_{3,P}$ and $imp_{3,T}$ are both injective. Hence $imp_3 \in \mathcal{I}$.

✓

4.1 Temporal Logic

This subsection coincides with Section 3.2 in [Pad06]. We use a notation closely related to standard linear time logics (LTL) as e.g. in [MP95] or [GV03]. For each net we assume a set of atomic propositions AP over the markings of the net. For a marking $m \in P^\oplus$ the satisfaction of a atomic proposition is given if the proposition \mathbf{p} is true for m .

A LTL formula is an element of the language

$$\mathbf{f} := \mathbf{p} \mid \neg \mathbf{f} \mid \mathbf{f} \wedge \mathbf{f} \mid \mathbf{X} \mathbf{f} \mid \mathbf{f} \mathbf{U} \mathbf{f}$$

constructed out of atomic propositions \mathbf{p} to which boolean connections \neg (negation) and \wedge (conjunction), as well as the temporal operators "until" \mathbf{U} and "next" \mathbf{X} are applied.

Since a LTL requires runs of a system we now define runs of a PT system (N, \hat{m}) as an infinite sequence of markings $\delta := m_0 \cdot m_1 \cdot m_2 \cdot \dots$ where $m_0 = \hat{m}$ is the initial marking. Either we have some $t \in T$ for each $i \geq 0$ so that $m_i[t > m_{i+1}]$ or we repeat the last marking, i.e. if there is no $t \in T$ such that $m_i[t > m_{i+1}]$ then $m_j = m_i$ for all $j > i$.

We assume a set of atomic propositions AP on markings, so that for each marking $\pi : P^\oplus \rightarrow 2^{AP}$ assigns truth values to the propositions. Thereby we have $\pi(m)(\mathbf{p}) = \text{true}$ for $\mathbf{p} \in AP$ and $m \in P^\oplus$ is denoted by $\mathbf{p} \in \pi(m)$.

Then we define inductively for formulas \mathbf{f} :

- for an atomic proposition $(\delta, j) \models \mathbf{p}$ iff $\mathbf{p} \in \pi(m_j)$ for $\mathbf{p} \in AP$
- for the boolean operators $(\delta, j) \models \neg \mathbf{f} \in AP$ iff not $(\delta, j) \models \mathbf{f}$
 $(\delta, j) \models \mathbf{f}_1 \wedge \mathbf{f}_2 \in AP$ iff $(\delta, j) \models \mathbf{f}_1$ and $(\delta, j) \models \mathbf{f}_2$

- for the until operator $(\delta, j) \models \mathbf{f}_1 \mathbf{U} \mathbf{f}_2$ iff

$$\begin{aligned} & \text{there is some } k \geq j \text{ with } (\delta, k) \models \mathbf{f}_2 \\ & \text{and for all } j \leq i \leq k \text{ holds } (\delta, i) \models \mathbf{f}_1 \end{aligned}$$
- for the next operator $(\delta, j) \models \mathbf{X} \mathbf{f}$ iff $(\delta, j+1) \models \mathbf{f}$

We abbreviate formulas using the usual boolean operators as they can be defined using the negation and the conjunction. Analogously we can define further temporal operators as "eventually" or "future" $\mathbf{F} \mathbf{f} := \text{true} \mathbf{U} \mathbf{f}$ and the operator "always" or "globally" $\mathbf{G} \mathbf{f} := \neg \mathbf{F} \neg \mathbf{f}$. The set of all LTL formulas with respect to the set of atomic propositions AP is denoted by \mathbb{F} .

A net system $(N, \widehat{m}) \models \mathbf{f}$ satisfies an LTL formula $\mathbf{f} \in \mathbb{F}$ if for all runs δ of (N, \widehat{m}) we have $(\delta, 0) \models \mathbf{f}$.

Definition 4.7 (Underlying logic for PT system components [Pad06])

The underlying temporal logic (\mathbb{F}, \models) over the net N consist of the formulas \mathbb{F} over the net N and the relation $\models_N \subseteq N \times \mathbb{F}$ where $N = \{(N, \widehat{m}) | \widehat{m} \in P^\oplus\}$ the set of all PT systems consisting of the net N and some initial marking $\widehat{m} \in P^\oplus$. \diamond

Next we define the translation of LTL formulas based on a mapping of the atomic propositions that is compatible with the mapping of the places and show then to be compatible with the composition of morphisms as required in Def. 3.7.

Definition 4.8 (Translation of a formula [Pad06])

Given PT systems (N_i, \widehat{m}_i) with atomic propositions AP_i and $\pi_i : P_i^\oplus \rightarrow AP_i$ for $1 \leq i \leq 2$, a morphism $h : PS_1 \rightarrow PS_2$, and a mapping of the atomic propositions $h_{AP} : AP_1 \rightarrow AP_2$ that is compatible with the mapping of the places i.e. $\pi_2 \circ h_P^\oplus = h_{AP} \circ \pi_1$, then we define $\mathfrak{T}_h : \mathbb{F}_{AP_1} \rightarrow \mathbb{F}_{AP_2}$ inductively:

- for atomic propositions $\mathfrak{T}_h(\mathbf{p}) := h_{AP}(\mathbf{p})$
- for the boolean operators $\mathfrak{T}_h(\neg \mathbf{f}) := \neg \mathfrak{T}_h(\mathbf{f})$
 $\mathfrak{T}_h(\mathbf{f}_1 \wedge \mathbf{f}_2) := \mathfrak{T}_h(\mathbf{f}_1) \wedge \mathfrak{T}_h(\mathbf{f}_2)$
- for the until operator $\mathfrak{T}_h(\mathbf{f}_1 \mathbf{U} \mathbf{f}_2) := \mathfrak{T}_h(\mathbf{f}_1) \mathbf{U} \mathfrak{T}_h(\mathbf{f}_2)$
- for the next operator $\mathfrak{T}_h(\mathbf{X} \mathbf{f}) := \mathbf{X} \mathfrak{T}_h(\mathbf{f})$
- for the eventually operator $\mathfrak{T}_h(\mathbf{F} \mathbf{f}) := \mathbf{F} \mathfrak{T}_h(\mathbf{f})$
- for the always operator $\mathfrak{T}_h(\mathbf{G} \mathbf{f}) := \mathbf{G} \mathfrak{T}_h(\mathbf{f})$

\diamond

Fact 4.9 (Composition of Translation [Pad06])

Given mappings of atomic propositions $h_{AP} : AP_1 \rightarrow AP_2$ and $g_{AP} : AP_2 \rightarrow AP_3$ compatible with $h : PS_1 \rightarrow PS_2$ and $g : PS_2 \rightarrow PS_3$, then we have $\mathfrak{T}_g \circ \mathfrak{T}_h = \mathfrak{T}_{g \circ h}$. \diamond

Result 4.10 (Component-based Verification for Split Imports)

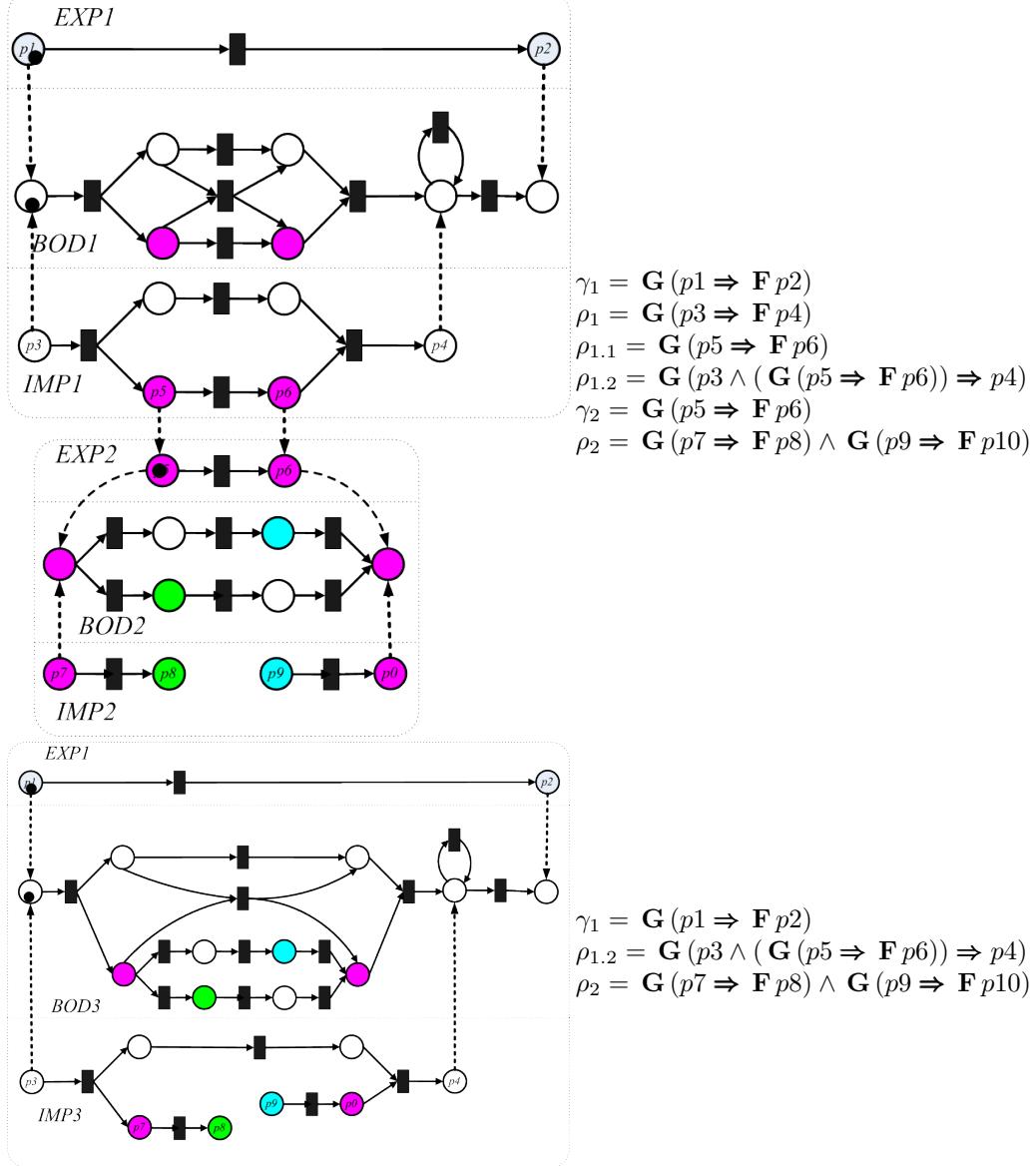
We now have

- PT system components with guarantees (see Def. 3.3), and
- partial composition propagating guarantees (see Def. 3.4).

\diamond

These results are due to Fact 4.2 that PT systems are an instantiation of the generic component framework in Section 2.

The following two figures extend the example from section 2 with the corresponding temporal formulas.



5 Conclusion

To summarize this report, a component is given formally by three specifications, the body specification, the import and the export interface. To express properties of components, an appropriate logic formalism has to be required that allows expressing the desired properties. A component is then equipped with two additional logic formulas that represent the import-export implication. The import assumptions describe in an abstract way the properties the underlying component needs to have to ensure the desired behavior. Then the export guarantees some property denoted by the export statement. Hierarchical composition allows concluding the import-export implication where the providing component's import assumption implies the requiring component's

export statement.

One important application of the instantiation of this generic frame work to PT net systems is the possibility of structuring nets hierarchically. This is crucial when modelling discrete systems in practical applications as discussed in [PKA08]. There we sketch the compositional verification based on this framework. The main issues when applying to the tool Netlab concern the feasibility in practice and the hiding of th temporal logic without loosing the possibility of verification.

References

- [BM07] J.F.K. Bowles and S. Moschouianis. Concurrent logic and automata combined: a semantics for components. In *Proc. of CONCUR 2006 - Foundations of Coordination Languages and Software Architectures (FOCLASA '06)*, pages 135–151. Electronic Notes in Theoretical Computer Science 175, Elsevier Science, 2007.
- [CBEO99] F. Cornelius, M. Baldamus, H. Ehrig, and F. Orejas. Abstract and behaviour module specifications. *Mathematical Structures in Computer Science*, 9:21–62, 1999.
- [dAH01] L. de Alfaro and T.A Henzinger. Interface automata. In *ESEC/FSE 01: Proceedings of the Joint 8th European Software Engineering Conference and 9th ACM SIGSOFT International Symposium on the Foundations of Software Engineering*, 2001.
- [EBK⁺05] H. Ehrig, B. Braatz, M. Klein, F. Orejas, S. Pérez, and E. Pino. Object-oriented connector-component architectures. In *Proc. Second International Workshop on Formal Foundations of Embedded Software and Component-based Software Architectures (FESCA 2005)*, Electronic Notes in Theoretical Computer Science 141, pages 123–151, 2005.
- [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 Verlag, Berlin, 1990.
- [EOB⁺02] H. Ehrig, F. Orejas, B. Braatz, M. Klein, and M. Piirainen. A Generic Component Concept for System Modeling. In *Proc. FASE 2002: Formal Aspects of Software Engineering*, Lecture Notes in Computer Science 2306 , pages 32–48. Springer Verlag, 2002.
- [EPB⁺04] H. Ehrig, J. Padberg, B. Braatz, M. Klein, F. Orejas, S. Pérez, and E. Pino. A generic framework for connector architectures based on components and transformations. In *Proc. First International Workshop on Formal Foundations of Embedded Software and Component-based Software Architectures (FESCA 2005)*, Electronic Notes in Theoretical Computer Science 108, pages 53–67, 2004.
- [GPB02] D. Giannakopoulou, C. Păsăreanu, and H. Barringer. Assumption generation for software component verification. In *Proceedings of ASE-2002: The 17th IEEE Conference on Automated Software Engineering*. IEEE CS Press, 2002.
- [GV03] C. Girault and R. Valk, editors. *Systems Engineering: a Guide to Model-ling, Verification and Applications*. Springer, 2003.
- [IWY00] Paola Inverardi, Alexander L. Wolf, and Daniel Yankelevich. Static checking of system behaviors using derived component assumptions. *ACM Trans. Softw. Eng. Methodol.*, 9(3):239–272, 2000.
- [JO99] Rosa M. Jiménez and Fernando Orejas. An algebraic framework for higher-order modules. In *FM'99 - Formal Methods, World Congress on Formal Methods in the Development of Computing Systems*, volume 1709 of *Lecture Notes in Computer Science*, pages 1778–1797. Springer, 1999.

- [MM90] J. Meseguer and U. Montanari. Petri Nets are Monoids. *Information and Computation*, 88(2):105–155, 1990.
- [MP95] Z. Manna and A. Pnueli. *Temporal Verification of Reactive Systems: Safety*. Springer Verlag, 1995.
- [Net07] Institut für Regelungstechnik, Rheinisch-Westfälische Technische Hochschule Aachen. *Petrinetz-Tool Netlab (Windows)*, 2007. <http://www.irt.rwth-aachen.de/typo3/index.php?id=101&L=0>.
- [OKA06] Ph. Orth, U. Küssel, and D. Abel. Netlab und Netlab Toolbox für Mat-lab/Simulink- ein Werkzeug zum Rapid Control Prototyping von Steuerungen mittels Petrinetzen. In *Tagungsband Entwurf komplexer Automatisierungssysteme EKA 2006*, pages 343–353. Institut für Regelungs- und Automatisierungstechnik TU Braunschweig, 2006.
- [Pad02] J. Padberg. Petri net modules. *Journal on Integrated Design and Process Technology*, 6(4):121–137, 2002.
- [Pad05] Julia Padberg. Integration of the generic component concepts for system modeling with adhesive HLR systems. *EATCS Bulletin*, 87:138–155, 2005.
- [Pad06] J. Padberg. Formal foundation for transformation-based approach to component verification. Technical Report 2006-12, Technische Universität Berlin,Fakultät IV, 2006.
- [PE05] J. Padberg and H. Ehrig. Petri net modules in the transformation-based component framework. *Journal of Logic and Algebraic Programming*, 67:198–225, 2005.
- [PEO07] J. Padberg, H. Ehrig, and F. Orejas. Towards component verification in the generic component framework. In J. Kuester-Filipe, I. Poernorno, and R. Reussner, editors, *Proc. Formal Foundations of Embedded Software and Component-Based Software Architectures (FESCA 07), Satellite Event of the European Joint Conferences on Theory and Practice of Software (ETAPS)*, Electronic Notes in Theoretical Computer Science, Amsterdam, 2007. Elsevier Science. to appear.
- [PKA08] J. Padberg, U. Küssel, and D. Abel. Hierarchical modelling and verification based on Petri net components with multiple import interfaces. In *IFAC Conference on Control Systems Design*, 2008. Accepted.
- [Sim99] M. Simeoni. *A Categorical Approach to Modularization of Graph Transformation Systems using Refinements*. PhD thesis, Università Roma La Sapienza, 1999.