Behavioural approach to modular system specification

H. Ehrig
M.P. Nivela
F. Orejas

Report LSI–88–24
Abstract: The algebraic approach to modular specification, as developed by the group of Berlin, is joined with the behavioural approach to algebraic specifications developed in Barcelona. In the joint approach module specifications are provided with a behavioural functorial semantics. As main results it is shown that basic interconnection mechanisms, like composition, union and actualization, are correctness preserving and compositional w.r.t. the behavioural semantics.

In practice also the case of view actualization is of great interest where non observable actual sorts can be bound to observable formal sorts. Also in this case, which requires behavioural view functors instead of forgetful functors, it is discussed how to extend the basic constructions and results.

Resum: L'enfoc algebraic de les especificacions modulars desenvolupades a Berlin es combina amb l'enfoc de comportament desenvolupat a Barcelona. En aquest enfoc conjunt les especificacions algebraiques són dotades de semàntica de comportament. Els resultats principals mostren que els mecanismes bàsics de connexió, com la composició, l'unió i la instanciació, preserven la correctesa i la composicionalitat respecte la semàntica de comportament.

A la pràctica és molt interessant el cas de la instanciació “vista”, en el que sorts observable en el paràmetre formal poden ser associats a sorts no observables del parmetre real. En aquest cas es necessari utilitzar funtors “vista” en lloc de funtors oblit i s'estudia com extendre els resultats bàsics.
Behavioural Approach to Modular System Specification

H. Ehrig *, Mª P. Nivela, F. Orejas
Universitat Politècnica de Catalunya
Barcelona, Spain
July, 1.988

Abstract
The algebraic approach to modular specifications as developed by the first author during the last five years is joined with the behavioural approach to algebraic specifications introduced recently by the last two authors. In the joint approach module specifications are provided with a behavioural functorial semantics. As main results we show that basic interconnection mechanisms, like composition, union and actualization are correctness preserving and compositional w.r.t the behavioural semantics. In practice also the case of view actualization is of great interest where non observable actual sorts can be bound to observable formal sorts. Also in this case, which requires behavioural view functors instead of forgetful functors, we discuss how to extend the basic constructions and results. Most constructions and results are illustrated by a behavioural module specification of a modular airport schedule system

.. Introduction

Module specifications were defined by Blum, Ehrig and Parisi-Presicce [BEP 87] as a concept for supporting the structured design of large specifications. In this framework, a module may be seen as a parameterized abstract data type extended by an import and an export interface. The export interface is the visible part describing what the module offers to a user. This allows to interconnect modules in a flexible manner and to build up modular software systems in stepwise style.

*on visit, permanent address Technical University of Berlin
In this section we will define the basic concepts concerning behavioural semantics of modules. In the first subsection we present the basic definitions and results about behavioural semantics: categories of behaviours, satisfaction relation associated to a set of equations, behaviour specification morphisms and its associated free and forgetful functors and pushouts results. In the second subsection the notion of behaviour module specification, its semantics and correctness concept is presented. Finally, the third subsection deals with the airport schedule system example.

2.1 General notion of behavioural semantics

2.1.1 Behavioural equivalence

Definition 2.1.1.1

A behaviour signature BSig is a triple BSig = (Obs, S, Σ) with Obs ⊆ S where Sig = (S, Σ) is a usual signature (see [EM 85]). The sorts in Obs are called observable sorts.

To every signature BSig, we may associate two categories of models: Alg(Sig) the well-known category of Sig-algebras and Sig-homomorphisms (see [EM 85]), and the one we will use for defining behavioural semantics, Beh(BSig) as introduced in [NO 87]. In this category models are Sig-algebras, as in Alg(Sig), but morphisms are different. To avoid confusion, from now on, morphisms in Alg(Sig) will be called Sig-homomorphisms, while morphisms in Beh(BSig) will be called Sig-behaviour morphisms.

Definition 2.1.1.2

Let A be a Sig-algebra. A computation (resp. an observable computation) over A is a term in \( T_Σ(A_{Obs}) \) (resp. \( T_Σ(A_{Obs})_S \) with \( s \in Obs \)).

Definition 2.1.1.3

Given two Sig-algebras A and B, a Sig-behaviour morphism \( f: A \to B \) is a family of functions \( f_s = \{f_s\}_{s \in Obs} \) such that for all observable computations \( t \in T_Σ(A_{Obs})_S \), \( s \in Obs \), the following equality holds

\[
 f_s (\varepsilon_A (t)) = \varepsilon_B (f^#_s (t))
\]

where \( f^#: T_Σ(A_{Obs}) \to T_Σ(B_{Obs}) \) is the unique Sig-homomorphism which extends \( f \) and \( \varepsilon_A \) is the evaluation function, i.e., the unique Sig-homomorphism extending the inclusion of \( A_{Obs} \) into A. Sig-algebras together with Sig-behaviour morphisms form the category Beh(BSig).

Remark 2.1.1.4

- It can be noted that if Obs coincides with S then Beh(BSig) is exactly the same as Alg(Sig).

- A Sig-behaviour morphism establishes a relationship between the observable computations and their results in A and B respectively. To every observable computation \( t \) over A corresponds the observable computation \( f^#(t) \) over B and to the result \( \varepsilon_A (t) \) of this computation over A corresponds the result \( \varepsilon_B (f^#(t)) \) in B. Thus an Obs-indexed family \( f = \{f_s\}_{s \in S} \) is a Sig-behaviour morphism if these observable computations over A yield in B the same value as in A, up to the transformation determined by \( f \). This means that the following diagram commutes:
for all sorts in Obs.

If the converse holds, that is, if all the observable computations over B yield in A the same value as in B up to the transformation determined by f, and f itself is a bijection, then A and B give the same answers to the same questions, that is, they show the same observable behaviour. Hence behavioural equivalence is characterized by isomorphism in the category Beh(BSig). In particular, isomorphism in Beh(BSig) coincides with the notion of behavioural equivalence from [MG 85, HWi85, SWi83, ST 85].

- A Sig-behaviour morphism f such that f_s is bijective for every s in Obs is Sig-behaviour isomorphism in the category Beh(BSig).

**Definition 2.1.1.5**

A and B are **behaviourally equivalent**, denoted $A \equiv_{BSig} B$, if there exists a Sig-behaviour isomorphism $f : A \rightarrow B$.

**Definition 2.1.1.6**

Let BSig = (Obs, S, $\Sigma$) be a behaviour signature, $X_{Obs}$ an Obs-indexed family of variables and z a variable symbol of sort s not belonging to X. A Sig-context over the sort s (resp. Sig-context for a Sig-algebra A over the sort s) is a term $c[z] \in T_\Sigma(X_{Obs} \cup \{z\})$, with $s' \in$ Obs (resp. $c_A[z] \in T_\Sigma(A_{Obs} \cup \{z\})$, with $s \in$ Obs).

Given a context $c[z]$ (resp. $c_A[z]$) and a term $t$ in $T_\Sigma(X_{Obs})$ (resp. in $T_\Sigma(A_{Obs})$), by $c(t)$ (resp. $c_A(t)$) we denote the application of the context over $t$, that is, $\sigma(c[z])$ (resp. $\sigma(c_A[z])$), where $\sigma$ is the assignment $\sigma : X_{Obs} \rightarrow T_\Sigma(X_{Obs})$ (resp. $\sigma : X_{A_{Obs}} \rightarrow T_\Sigma(X_{A_{Obs}})$) defined by $\sigma(z) = t$ and $\sigma(x) = x$ for every $x$ in $X_{Obs}$ (resp. $\sigma(z) = t$).

**Definition 2.1.1.7**

Let A be a Sig-algebra and e: $\lambda Y.t_1=t_2$ a Sig-equation of sort s. A behaviourally satisfies e, denoted $A \models_{BSig} e$, if A satisfies $\lambda X_{Obs} c[\overline{\sigma(t_1)}] = c[\overline{\sigma(t_2)}]$ for every Sig-context $c[z]$ over the sort s and every assignment $\sigma : Y \rightarrow T_\Sigma(X_{Obs})$.

**Remark 2.1.1.8**

- If we consider contexts as observations, it would seem quite natural to assert that "an algebra A behaviourally satisfies the equation e if A satisfies $c[t_1] = c[t_2]$ for every Sig-context $c[y]$ over the sort s", that is, A satisfies every kind of observable consequence of e. In this way, an equation of observable sort would be satisfied in the classical sense and an equation of non observable sort would be satisfied up to observable contexts. This is the notion of behavioural satisfaction defined in [RE 84]. However, in our framework, this definition is not appropriate because the replacement of the variables in the equation by "non observable inputs " (more specifically, by non observable junk) can cause that behavioural equivalent algebras do not behaviourally satisfy the same equations. This problem is not present in [RE 84] because of his different notion of behavioural equivalence. So we will only allow to substitute variables by values finitely generated from the observable ones. This is coherent with the intuitive idea.
that "we do not care about the nonobservable part".

- From now on, in order to enhance readability, we will omit the explicit quantification of variables in equations, though technically needed [GM 81].

**Definition 2.1.1.9**

A behaviour specification \( \text{BSPEC} \) is a 4-tuple, \( \text{BSPEC} = (\text{Obs}, S, \Sigma, E) \), where \( \text{BSig} = (\text{Obs}, S, \Sigma) \) is a behaviour signature and \( E \) a set of \( \text{Sig}=(S, \Sigma) \)-equations. \( \text{Beh(BSPEC)} \) is the full subcategory of \( \text{Beh(BSig)} \) of all \( \text{Sig} \)-algebras which behaviourally satisfy the equations in \( E \).

In what follows we will shortly write \( \text{BSPEC} = (\text{Obs}, \text{SPEC}) \) with \( \text{SPEC} = (S, \Sigma, E) \) and \( \text{Obs} \subseteq S \).

### 2.1.2 Behaviour specification morphisms, parameterized behaviour specifications, free constructions and pushouts

**Definition 2.1.2.1**

Let \( \text{BSPECi} = (\text{Obsi}, \text{SPECi}) \), \( i = 1, 2 \) be two behaviour specifications with \( \text{SPECi} = (\text{Si}, \Sigma_i, E_i) \). A behaviour specification morphism \( h : \text{BSPEC1} \rightarrow \text{BSPEC2} \) is a specification morphism \( h : \text{SPEC1} \rightarrow \text{SPEC2} \) (i.e. \( h \) is a signature morphism with \( E_2 \vdash h(E_1) \)) such that \( h(\text{Obs1}) \subseteq \text{Obs2} \) and \( h(\text{S1-Obs1}) \subseteq \text{S2-Obs2} \) (i.e. observable and non observable sorts are preserved).

**Remark 2.1.2.2**

- To a behaviour specification morphism \( h : \text{BSPEC1} \rightarrow \text{BSPEC2} \) it can be associated a forgetful functor \( \text{BU}_h : \text{Beh(BSPEC2)} \rightarrow \text{Beh(BSPEC1)} \) defined as usual. Due to \( h(\text{Obs1}) \subseteq \text{Obs2} \) it holds that if \( A_2 \) and \( A_2' \) are two algebras in \( \text{Beh(BSPEC2)} \) such that \( A_2 \equiv_{\text{BSPEC2}} A_2' \) then \( \text{BU}_h(A_2) \equiv_{\text{BSPEC2}} \text{BU}_h(A_2') \).

- For the existence of free and forgetful functors is not needed that \( h(\text{S1-Obs1}) \subseteq \text{S2-Obs2} \) [NO 87].

Associated to a behaviour specification morphism \( h \) there is a free functor, \( \text{BFree}_h \), left adjoint to the forgetful functor \( \text{BU}_h \).

**Theorem 2.1.2.3**

Let \( \text{BSPECi} = (\text{Obsi}, \text{SPECi}), \text{SPECi} = (\text{Si}, \Sigma_i, E_i), i = 1, 2 \) be two behaviour specifications and \( h : \text{BSPEC1} \rightarrow \text{BSPEC2} \) a behaviour specification morphism. For every algebra \( A \) in \( \text{Beh(BSPEC1)} \), the algebra defined by

\[
\text{BFree}_h(A) = T_{\text{S2}}(A_{\text{Obs1}}/\equiv_{\text{obs}(A)}+E_2)
\]

(interpreting the values \( a \in A_{\text{S1}}, s \in \text{Obs1} \), as values of sort \( h(s) \)) is a free construction over \( A \) with respect to \( \text{U}_h \) which extends to a free functor \( \text{BFree}_h : \text{Beh(BSPEC1)} \rightarrow \text{Beh(BSPEC2)} \).

**Remark 2.1.2.4**

- Let us observe that if \( \text{Obs1} = S_1 \) then for every \( \text{SPEC1} \)-algebra \( A \), \( \text{BFree}_h(A) \) is the usual free construction.

- Even if \( h \) is not preserving non observable sorts, i.e. \( h(\text{S1-Obs1}) \nsubseteq \text{S2-Obs2} \), the forgetful and
free functors $U_h$ and $\text{BFree}_h$ exist [NO 37].

- The behavioural equivalence relation may be extended uniformly from algebras to functors, that is, behavioural equivalence of functors coincides with natural isomorphism. So, if $F$ and $F'$ are two functors from $\text{Beh}(\text{BSPEC1})$ to $\text{Beh}(\text{BSPEC2})$ we will say that $F$ and $F'$ are behaviourally equivalent if they are naturally isomorphic and we will denote that by $F \equiv F'$. Therefore, we immediately have that any functor behaviourally equivalent to a free functor is also free.

- Moreover, a functor $F : \text{Beh}(\text{BSPEC1}) \rightarrow \text{Beh}(\text{BSPEC2})$ is persistent with respect $U : \text{Beh}(\text{BSPEC2}) \rightarrow \text{Beh}(\text{BSPEC1})$ if $U \cdot F \equiv \text{Id}_{\text{Beh}(\text{BSPEC1})}$.

**Definition 2.1.2.5**

Let $\text{BSPEC}i = (\text{Obs}i, \text{SPEC}i)$, $\text{SPEC}i = (S_i, \Sigma_i, E_i)$, $\text{Obs}i \subseteq S_i$, $i = 1, 2$ be two behaviour specifications. A **parameterized behaviour specification** $\text{BPSPEC}$ is a pair of behaviour specifications $\text{BPSPEC} = (\text{BSPEC1}, \text{BSPEC2})$ such that $\text{Obs}1 = S1$ and $\text{SP}1 \subseteq \text{SP}2$. $\text{BSPEC}1$ is called formal parameter specification and $\text{BSPEC}2$ the target specification. The behavioural semantics of $\text{BPSPEC}$ is the (class of) free functor(s) associated to the inclusion specification morphism.

Let Set, SPEC and BSPEC be the categories of sets and maps, specifications and specification morphisms and behaviour specifications and behaviour specification morphisms respectively. We show that we have pushouts in the category BSPEC, similar to those in SPEC.

**Proposition 2.1.2.6**

Let $\text{BSPEC}i = (\text{Obs}i, \text{SP}i)$, $i = 1, 2$ be two behaviour specifications and $i_1 : \text{BSPEC}1 \rightarrow \text{BSPEC}2$, $h_1 : \text{BSPEC}1 \rightarrow \text{BSPEC}3$ two behaviour specification morphisms. The behaviour specification $\text{BSPEC}4 = (\text{Obs}4, \text{SP}4)$, defined by $\text{Obs}4$ being the pushout of $i_1'$ and $h_1'$ in the category Sets, $\text{SP}4$ the pushout of $i_1$ and $h_1$ in the category SPEC, together with the behaviour specification morphisms $h_2 : \text{BSPEC}2 \rightarrow \text{BSPEC}4$ and $i_2 : \text{BSPEC}3 \rightarrow \text{BSPEC}4$ defined by the corresponding pushouts diagrams below,

$$
\begin{array}{ccc}
\text{SP1} & \xrightarrow{i_1} & \text{SP2} \\
| & h_1 | & | \\
\text{SP3} & \xrightarrow{i_2} & \text{SP4} \\
\end{array}
\quad
\begin{array}{ccc}
\text{Obs1} & \xrightarrow{i_1'} & \text{Obs2} \\
| & h_1' | & | \\
\text{Obs3} & \xrightarrow{i_2'} & \text{Obs4} \\
\end{array}
$$

is a pushout on BSPEC, where $i_1' = i_1|_{\text{Obs}1}$, $h_1' = h_1|_{\text{Obs}1}$, $i_2' = i_2|_{\text{Obs}2}$ and $h_2' = h_2|_{\text{Obs}2}$ are the restrictions of $i_1$, $h_1$, $i_2$ and $h_2$ to the their respective observable sorts.

**Definition 2.1.2.7**

A pushout diagram

$$
\begin{array}{ccc}
\text{BSPEC1} & \xrightarrow{i_1} & \text{BSPEC2} \\
| & h_1 | & | \\
\text{BSPEC3} & \xrightarrow{i_2} & \text{BSPEC4} \\
\end{array}
$$

satisfies the observation preserving property if for every minimal observable computation $t \in T_{\Sigma 4}(X_{\text{Obs}4})$, (i.e. no subterm of $t$, different from a variable, is an observable computation) and
every $s \in S4 - \text{Obs4}$ being the sort of a non observable subterm of $t$ it holds that $s \in S4 - i2 \cdot h1(S1)$ or, equivalently, $t \in T_{i2}(\Sigma3) \cup T_{h2}(\Sigma3)$.

Remarks 2.1.2.3

- Observation preserving will be needed to obtain most of our results. In fact, if a diagram does not satisfy it then there is no Amalgamation Lemma associated to it.
- The problem when this property is not satisfied is that BSPEC3 and BSPEC2 may add more ways of "observing" the values of the non observable sorts of BSPEC1. This makes incompatible the semantics of (BSPEC3, BSPEC4) and (BSPEC1, BSPEC2).
- If Obs1 = S1 then the property trivially holds. Under this assumption all results about parameter passing were obtained in [NO 87].

2.2 General notion of behaviour module specification

The concept of a behaviour module is an extension of the module concept in [BEP 87] in which observable sorts are distinguished in each part in order to use the concepts of behavioural specifications in the sense of [NO 87].

Definition 2.2.1

A behaviour module specification $\text{BMOD}$ is a 8-tuple

$$\text{BMOD}= (\text{BPAR}, \text{BEXP}, \text{BIMP}, \text{BBOD}, e, s, i, v)$$

where:

a) $\text{BPAR} = (\text{ObsPAR}, \text{PAR})$ is called behaviour parameter specification  
$\text{BEXP} = (\text{ObsEXP}, \text{EXP})$ is called behaviour export interface specification  
$\text{BIMP} = (\text{ObsIMP}, \text{IMP})$ is called behaviour import interface specification  
$\text{BBOD} = (\text{ObsBOD}, \text{BOD})$ is called behaviour body specification

b) $e, s, i, v$ are behaviour specification morphism such that $v \cdot e = s \cdot i$, that is, the following diagram commutes

\[
\begin{array}{ccc}
\text{BPAR} & \xrightarrow{e} & \text{BEXP} \\
\downarrow{i} & & \downarrow{v} \\
\text{BIMP} & \xrightarrow{s} & \text{BBOD}
\end{array}
\]

We will just write $\text{BMOD} = (\text{BPAR}, \text{BEXP}, \text{BIMP}, \text{BBOD})$ if the behaviour specification morphisms $e, s, i$ and $v$ are clear in the context.

Forgetting the distinction of observable sorts we obtain a usual module specification $\text{MOD} = (\text{PAR}, \text{EXP}, \text{IMP}, \text{BOD}, e, s, i, v)$ in the sense of [BEP 87].

Definition 2.2.2

Let $\text{BMOD} = (\text{BPAR}, \text{BEXP}, \text{BIMP}, \text{BBOD}, e, s, i, v)$ be a behaviour module specification. The behavioural semantics of $\text{BMOD}$ is the functor $\text{BSEM} = U_v \cdot \text{BFree}_s$ where $\text{BFree}_s$ is the free
functor w.r.t. the forgetful functor $U_S$ (see 2.1.2.3).

Definition 2.2.3

A behaviour module specification $BMOD = (BPAR, BEXP, BIMP, BBOD, e, s, i, v)$ is behaviourally correct if $BFree_s$ is strongly persistent, that is, $BU_S \cdot BFree_s = \text{Id}_{\text{Beh}(BIMP)}$.

Proposition 2.2.4

If $BMOD$ is a correct behaviour module specification then $BU_e \cdot BSEM = BU_i$.

Proof

Due to that $v \cdot e = s \cdot i$ we have $BU_e \cdot BSEM = BU_e \cdot BU_v = BFree_s = BU_i \cdot BU_S \cdot BFree_s$.

Moreover, by persistency of $BFree_s$, we have $BU_e \cdot BSEM = BU_i$.

Definition 2.2.5

Given two behaviour module specifications $BMOD_j = (BPAR_j, BEXP_j, BIMP_j, BBOD_j), j = 1, 2$, a behaviour module specification morphism $f : BMOD1 \rightarrow BMOD2$ is a 4-tuple $f = (f_P, f_E, f_I, f_B)$ of behaviour specification morphisms such that the following diagram commutes

![Diagram](image)

Definition 2.2.6

A behaviour module specification morphism $f : BMOD1 \rightarrow BMOD2$ is coherent if the semantics $BSEM1$ and $BSEM2$ of $BMOD1$ and $BMOD2$ are compatible with the corresponding forgetful functors up to behavioural equivalence, i.e.

$$BSEM1 \cdot BU_{f_I} = BU_{f_E} \cdot BSEM2$$

2.3 Example

The airport schedule system specified in [EW 86] can be studied under the behavioural approach establishing the following observable sorts in each part of the module.

$$aps\text{-bmod} = (\text{aps-par, aps-exp, aps-imp, aps-bod})$$

$$\text{aps-par} = \text{Bool} +$$

$$\text{hspec}$$

$$\text{sorts obs}$$

flight# {flight number}, dest {destination}, dep {departure},
plane# {plane number}, type, seats {number of seats}
Airport schedules are constructed by means of a hidden function "tup" of the aps-bod part which builds airport schedules as tuples of flight schedules (fs) and plane schedules (ps). This function is hidden because in general it constructs inconsistent elements.

The sorts "flight#", "dest", "dep", "plane#", "type", "seats", "fs" and "ps" are considered as observable and the sort "aps" as non observable. The behavioural semantics of the aps-bmod is the functor BSEM which transforms aps-imp-algebras into aps-exp-algebras. Given an aps-imp-algebra A it constructs the aps-export-algebra U • Free(A). Due to that the sort "aps" is non observable it is not necessary, as in the usual semantics of modules [BEP 87], to remove the junk (i.e. inconsistent airport schedules) that can be generated in the body, because the two algebras U • Free(A) and Restr • U • Free(A) are behaviourally equivalent, where Restr is the restriction functor of [BEP 87].

3. Composition, union and actualization

In this section we introduce as basic interconnection mechanisms for behavioural module specifications the operations composition, union and actualization. They are extension of the corresponding operations for usual module specifications as introduced in [BEP 87]. We are able to show that also the main results concerning correctness and compositionality of these operations can be extended using a new Amalgamation and Extension Lemma for the behavioural case with behaviour specification morphisms, provided that the corresponding constructions are behavioural consistent.
Definition 3.1

The composition of two behaviour module specifications $\text{BMOD}_i = (\text{BPAR}_i, \text{BEXP}_i, \text{BIMP}_i, \text{BBOD}_i)$, $i = 1, 2$, via a pair $h = (h_1, h_2)$ of behaviour specification morphisms making diagram (3) below commutative, is the behaviour module specification $\text{BMOD}_3 = (\text{BPAR}_3, \text{BEXP}_3, \text{BIMP}_3, \text{BBOD}_3)$ given by the outer square in

$$
\begin{array}{ccc}
\text{BPAR}_1 & \xrightarrow{id} & \text{BPAR}_1 \\
\downarrow & & \downarrow \\
\text{BIMP}_1 & \xrightarrow{(1)} & \text{BBOD}_1 \\
\text{BPAR}_2 & \xrightarrow{(2)} & \text{BEXP}_2 \\
\downarrow & & \downarrow \\
\text{BIMP}_2 & \xrightarrow{(4)} & \text{BBOD}_2 \\
\end{array}
$$

with $\text{BPAR}_3 = \text{BPAR}_1$, $\text{BEXP}_3 = \text{BEXP}_1$, $\text{BIMP}_3 = \text{BIMP}_2$ and $\text{BBOD}_3$ the pushout object in (4), written $\text{BMOD}_3 = \text{BMOD}_2 \uplus_h \text{BMOD}_1$.

The composition is called \textit{behavioural consistent} if pushout (4) satisfies the observation preserving property (see 2.2.2.7).

Theorem 3.2 (Correctness and compositionality of composition)

Given correct behaviour module specifications $\text{BMOD}_1$ and $\text{BMOD}_2$ and a pair $h = (h_1, h_2)$ such that the composition $\text{BMOD}_3 = \text{BMOD}_2 \uplus_h \text{BMOD}_1$ is defined and behavioural consistent, we have

1. Correctness $\text{BMOD}_3$ is correct
2. Compositionality $\text{BSEM}_3 = \text{BSEM}_1 \cdot \text{BU}_{h_1} \cdot \text{BSEM}_2$
   where $\text{BSEM}_j$, $j = 1, 2, 3$ is the behavioural semantics of $\text{BMOD}_j$

Proof idea

Extend the proof of theorem 3.3 in [BEP 87] to the behavioural case using the Behavioural Extension Lemma (see Appendix) applied to pushout (4) in definition 3.1 satisfying the observation preserving property (see 2.2.2.7).

Definition 3.3

The actualization of a behaviour module specification $\text{BMOD} = (\text{BPAR}, \text{BEXP}, \text{BIMP}, \text{BBOD})$ by a parameterized behaviour specification $\text{BPSPEC}_1 = (\text{BPAR}_1, \text{BACT})$ via a behaviour specification morphism $h : \text{BPAR} \to \text{BACT}$ is the behaviour module specification $\text{BMOD}_1 = (\text{BPAR}_1, \text{BEXP}_1, \text{BIMP}_1, \text{BBOD}_1)$ given by the outer front square in

$$
\begin{array}{ccc}
\text{BPAR} & \xrightarrow{\text{i}} & \text{BACT} \\
\downarrow & & \downarrow \\
\text{BIMP} & \xrightarrow{\text{h}_0} & \text{BBOD} \\
\text{BPAR}_1 & \xrightarrow{\text{in}} & \text{BEXP}_1 \\
\downarrow & & \downarrow \\
\text{BIMP}_1 & \xrightarrow{id} & \text{BIMP}_1 \\
\end{array}
$$
where BEXP1, BIMP1 and BBOD1 are constructed as pushouts in the top, left and bottom part respectively, written BMOD1 = BMODh(BPSPEC1).

The actualization is called behavioural consistent if the bottom pushout satisfies the observation preserving property (see 2.2.2.7).

**Theorem 3.4** (Correctness and compositionality of actualization )

Given a correct behaviour module specification BMOD, a parameterized behaviour specification BPSPEC1 and a morphism h such that the actualization BMOD1 = BMODh(BPSPEC1) is defined and behavioural consistent, we have

1. Correctness BMOD1 is correct
2. Compositionality BSEM1(I ⊗p F(P1)) = BSEM(I) ⊗p F(P1)

where BSEM, BSEM1 and F are the behavioural semantics of BMOD, BMOD1 and BPSPEC1 respectively. The operation ⊗ denotes the behavioural amalgamated sum of a BIMP-algebra I, BPAR-algebra P and BACT-algebra F(P1) on the left hand side defining a BIMP1-algebra I ⊗p F(P1), and similarly a BEXP1-algebra on the right hand side.

**Proof idea**

 Extend the corresponding proof in [PP 87] for the non behavioural case using the Behavioural Extension and Amalgamation Lemma (see Appendix) where the pushouts in diagram 3.3 satisfy the observation preserving property by behavioural consistency of actualization.

**Definition 3.5**

Given behaviour module specifications BMODj, j= 0, 1, 2, and two behaviour module specification morphisms fj : BMOD0 → BMODj, j = 1, 2, the union of MOD1 and MOD2 via MOD0 and (f1, f2) is given by the pushout BMOD3, written BMOD3 = BMOD1 +BMOD0 BMOD2, in the category of behaviour module specifications and behaviour module specification morphisms.

The union is called **behavioural consistent** if the pushout satisfies the observation preserving property in all componenets (see 2.1.2.7)

```
     f1
    /   \
   BMOD0   f2
     \   /  \
      g1   g2
    BMOD1 --- BMOD2
     \       /  \
      g1     g2
    BMOD3 
```

**Remark 3.6**

* Similar to [PP 87] actualization can be considered as a special case of union also for behaviour module specifications.

**Theorem 3.7** (Correctness and compositionality of union)

Given correct behaviour module specifications BMODj, j = 0, 1, 2, and coherent behaviour module specification morphisms fj, such that the union BMOD3 = BMOD1 +BMOD0 BMOD2 is behavioural consistent, then we have
1. Correctness $BMOD_3$ is correct
2. Compositionality $BSEM_3 = BSEM_1 \uplus BSEM_0 \uplus BSEM_2$
\[\text{where } BSEM_j \text{ is the behavioural semantics of } BMOD_j \text{ for } j = 0, 1, 2, 3 \text{ and } \uplus \text{ the behavioural amalgamated sum of functors}\]

**Proof idea**

Extend the proof of theorem 4.12 in [BEP 87] to the behavioural case using the Behavioural Extension Lemma (see Appendix) where all the pushouts satisfy the observation preserving property.

**Example 3.8 (Interconnection of behaviour modules in the airport schedule system)**

The example in section 2.3 can be enriched considering the following behaviour modules

bool-bmod, fs-bmod, ps-bmod:

\[
\begin{align*}
\text{bool-bmod} &= \{\text{bool-par, bool-exp, bool-imp, bool-bod}\} \\
\text{bool-par} &= \langle \emptyset, \emptyset, \emptyset, \emptyset \rangle \\
\text{bool-exp} &= \text{hspec} \\
&\quad \text{sorts} \quad \text{obs} \quad \text{bool} \\
&\quad \text{ops} \quad \text{true, false} : \rightarrow \text{bool} \\
&\quad \text{not} : \rightarrow \text{bool} \\
&\quad \text{and, or} : \text{bool bool} \rightarrow \text{bool} \\
&\quad \text{cons} \quad \{\text{the usual ones}\} \\
\text{end hspec}
\end{align*}
\]

\[
\begin{align*}
\text{fs-bmod} &= \{\text{fs-par, fs-exp, fs-imp, fs-bod}\} \\
\text{fs-par} &= \text{bool-exp} + \\
&\quad \text{hspec} \\
&\quad \text{sorts} \quad \text{obs} \quad \text{flight#, dest, dep} \\
&\quad \text{ops} \quad \text{eq} : \text{flight# flight#} \rightarrow \text{bool} \\
&\quad \text{cons} \quad \text{fn} : \text{flight#} \\
&\quad \text{eq} (\text{fn, fn}) = \text{true} \\
&\quad \text{hcons} \\
\text{end hspec}
\end{align*}
\]

\[
\begin{align*}
\text{ps-bmod} &= \{\text{ps-par, ps-exp, ps-imp, ps-bod}\} \\
\text{ps-par} &= \text{bool-exp} + \\
&\quad \text{hspec} \\
&\quad \text{sorts} \quad \text{obs} \quad \text{plane#, type, seats} \\
&\quad \text{ops} \quad \text{eq} : \text{plane# plane#} \rightarrow \text{bool} \\
&\quad \text{oper} \quad \text{ps} : \text{plane#} \\
&\quad \text{eq} (\text{ps}, \text{ps}) = \text{true} \\
&\quad \text{hcons} \\
\text{end hspec}
\end{align*}
\]

The inclusions $i_1 : \text{bool-bmod} \rightarrow \text{fs-bmod}$ and $i_2 : \text{bool-bmod} \rightarrow \text{ps-bmod}$ are coherent behaviour specification morphisms such that the module $\text{fsps-bmod}$ defined as the union

\[
\text{fsps-bmod} = \text{fs-bmod} \circ \text{bool-bmod \ ps-bmod}
\]
is well defined, behavioural consistent and correct (see thm. 3.6). Now we are able to define the composition

\[
\text{aps-system-bmod} = \text{aps-bmod} \circ h \text{fsps-bmod}
\]

where $h = (id, id)$ is a pair of identity morphisms. The $\text{aps-system-bmod}$ module is behavioural coherent and by theorem 3.2 is again correct and its behavioural semantics can be composed by those of
4. Composition and actualization with view morphisms

The operations of composition and actualization, as defined in the previous section are sometimes insufficient to work with module specifications with true behavioural semantics. For instance, the fact that the fitting morphism used for actualization would be a behaviour specification morphism, together with the fact that all parameter sorts are observable, would impose, to satisfy the behavioural consistency property of actualization, that all sorts of the actual parameter bounded to sorts of the module parameter should be observable, since behaviour specification morphism preserve observable sorts.

This, for example, would mean that it would not be possible to instantiate the \texttt{aps-system-bmod} module with a flight number which is a sequence of characters, if the sort sequence is not observable.

A similar situation occurs when trying to compose two modules. If for semantic reasons we have defined an import sort of the first module to be observable, the corresponding output sort of the second module would need to be also observable, if we want to satisfy the behavioural consistency property for composition. For instance, if in the \texttt{aps-bmod} module we have declared the sort \texttt{fs} to be observable, since otherwise the semantics of import algebras would be too poor because there are not enough observations on this sort, then it would be impossible to compose \texttt{aps-bmod} with \texttt{fs-bmod} if in the latter \texttt{fs} has been declared non observable.

To overcome this problem, it is sensible to allow working with a more general notion of behaviour specification morphism, namely what we have called \textit{view specification morphisms}.

\textbf{Definition 4.1}

Let \( \text{BSPECi}=(\text{Obsi,SPECi}) \) \( i = 1, 2 \) be two behaviour specifications with \( \text{SPECi} = (S_i, \Sigma_i, E_i) \). A view specification morphism \( h : \text{BSPEC1} \rightarrow \text{BSPEC2} \) is a specification morphism \( h : \text{SPEC1} \rightarrow \text{SPEC2} \) such that \( h(\text{S1-Obs1}) \subseteq \text{S2-Obs2} \).

\textbf{Remark 4.2}

- A view specification morphism not necessarily satisfies \( h(\text{Obs1}) \subseteq \text{Obs2} \) as required for behaviour specification morphisms.

- Let us observe that if \( h \) is a view specification morphism then it has no sense to consider \( U_h \) because in \( \text{Obs2} \) there can be "less" observable sorts than in \( \text{Obs1} \). This means that when "forgetting" a \( \text{BSPEC2}\)-behaviour morphism \( f = \{ f_s : A_2 s \rightarrow A'_2 s \} s \in \text{Obs2} \) there can exist some sort \( s \in \text{Obs1} \) such that \( h(s) \notin \text{Obs2} \) and, therefore, \( f_{h(s)} \) is not defined. Passing from \( \text{BSPEC2} \) to \( \text{BSPEC1} \) behaviours can be done by a functor \( \text{View}_{h} \), called view functor, which builds up a \( \text{BSPEC1}\)-behaviour morphism from a \( \text{BSPEC2}\)-behaviour morphism and describes how the models of the behaviour of an algebra \( A_2 \) in \( \text{Beh} (\text{BSPEC2}) \) are "seen" from the \( \text{BSPEC1} \) "point of view". First of all a special realization of the behaviour of an algebra \( A_2 \) in \( \text{Beh} (\text{SPEC2}) \) is constructed, that is, an algebra behaviourally equivalent to \( A_2 \) but in the category \( \text{Alg} (\text{SPEC2}* + h(E1)) \) defined below, in such a way that \( \text{BSPEC2}\)-behaviour morphisms can be extended to usual \( \text{SPEC2}* + h(E1) \) homomorphisms. Secondly, once we are in \( \text{Alg} (\text{SPEC2}* + h(E1)) \), a forgetful functor from \( \text{Alg} (\text{SPEC2}* + h(E1)) \) to \( \text{Beh} (\text{BSPEC1}) \) is applied.

- Pushouts for this kind of morphisms would be defined in a similar way as for behaviour specification morphisms. However, in this section we will only work with "mixed" diagrams, i.e. diagrams in which two morphisms are behaviour specification and the other two are view specification.
morphisms. Formally, this poses no problem since the formers are a special case of the latters. Also, the observation preserving property can be defined accordingly.

**Definition 4.3**

Let $BSPEC = (Obs, SPEC)$ be a behaviour specification with $SPEC = (S, \Sigma, E)$. The specification behaviourally derived from $BSPEC$ is $SPEC^* = (S, \Sigma, E^*)$ where

$$E^* = \{ t_1 = t_2 \mid t_1, t_2 \in T_\Sigma(X_{Obs})_S, s \in Obs, E \vdash t_1 = t_2 \}$$

is the set of all equations of observable sort over observable variables which may be deduced from $E$.

**Remark 4.4**

- The category $Beh(BSPEC)$ has the same objects as $Alg(SPEC^*)$ but behaviour morphisms are in a sense "weaker" than homomorphisms: every $\Sigma$-homomorphism may be seen as a behaviour morphism forgetting about the functions defined in non observable sorts. That allows to consider a forgetful functor

$$UBSPEC : Alg(SPEC^*) \rightarrow Beh(BSPEC)$$

defined by $UBSPEC(A) = A$ for every $A$ in $Alg(SPEC^*)$ and to every $\Sigma$-homomorphism $f$ corresponds $UBSPEC(f) = f_{Obs}$, that is the same $f$ viewed as a $\Sigma$-behaviour morphism by its restriction to the observable sorts.

**Definition 4.5**

Let $BSPEC = (Obs, SPEC)$ be a behaviour specification with $SPEC = (S, \Sigma, E)$, $E'$ a set of equations such that $SPEC \vdash E'$ and $A$ in $Beh(BSPEC)$. The $E'$-realization of $A$ is the algebra

$$E'^{-}\text{R}(A) = T_\Sigma(A_{Obs}) / \equiv_{eobs(A)+E'}$$

**Proposition 4.6**

Let $BSPEC = (Obs, SPEC)$ be a behaviour specification with $SPEC = (S, \Sigma, E)$, $E'$ a set of equations such that $SPEC \vdash E'$ and $A$ in $Beh(BSPEC)$. Then we have:

- The $E'$-realization of $A$ is behaviourally equivalent to $A$

- The construction of this $E'$-realization extends to a functor $E'^{-}\text{R} : Beh(BSPEC) \rightarrow Alg(SPEC^*+E')$, which is left adjoint to the forgetful functor $U : Alg(SPEC^*+E') \rightarrow Beh(BSPEC)$, defined as the composition of the forgetful functors $U1 : Alg(SPEC^*+E') \rightarrow Alg(SPEC^*)$ and $UBSPEC : Alg(SPEC^*) \rightarrow Beh(BSPEC)$.

**Definition 4.7**

Let $BSPEC_i = (Obs_i, SPEC_i)$ with $SPEC_i = (Si, \Sigma_i, E_i)$ $i = 1, 2$, be two behaviour specifications and $h : BSPEC1 \rightarrow BSPEC2$ a view specification morphism. The functor $View_h : Beh(BSPEC2) \rightarrow Beh(BSPEC1)$, called view functor associated to $h$, is defined as the composition of the functors $h(E1)^{-}\text{R} : Beh(BSPEC2) \rightarrow Alg(SPEC2^*+h(E))$ and $U : Alg(SPEC2^*+h(E)) \rightarrow Beh(BSPEC)$, i.e.

$$View_h = U \circ h(E1)^{-}\text{R}.$$

In what follows, we will define the operation of view composition in terms of view specification
morphism and present the usual correctness and compositionality theorems. However, the results have been harder to obtain since there is no amalgamated sum associated to pushout diagrams involving view morphisms. The only result were able to obtain and use has been a View Extension Lemma (see Appendix).

We could have provided a similar definition for view actualization, showing without problems the correctness of the operation. However, the standard prove of compositionality for the operation of actualization makes use of the amalgamation lemma that, as it was said above, does not hold for this kind of diagrams. However, it seems possible to prove compositionality using just the extension lemma.

With respect to union, there does not need to be a methodological justification of its definition via view morphisms. However, if there were we would have the same problems concerning the proof of compositionality

**Definition 4.8**

The view composition of two behaviour module specifications $BMOD_j = (BPAR_j, BEXP_j, BIMP_j, BBOD_j, ej, sj, ij, vj)$, $j = 1, 2$, via a pair $h = (h_1, h_2)$ is defined in the same way as composition in 3.1 up to the following modifications:

- $h_1$ and $h_2$ are view specification morphisms
- behavioural consistency includes the property that $e_3$ becomes a behaviour specification morphism

**Theorem 4.8 (Correctness and compositionality of view composition)**

Given correct behaviour module specifications $BMOD_1$ and $BMOD_2$ and a pair $h = (h_1, h_2)$ of view specification morphisms such that the view composition $BMOD_3 = BMOD_2 \circ_h BMOD_1$ is defined and behavioural consistent, we have

1. **Correctness** $BMOD_3$ is correct
2. **Compositionality** $BSEM_3 = BSEM_1 \circ_{BSEM_1} \circ_{BSEM_2}$

where $BSEM_j, j = 1, 2, 3$ is the behavioural semantics of $BMOD_j$.

**Proof idea**

See proof idea of theorem 3.2 using View Extension Lemma instead of Behaviour Extension Lemma.

5. References


Appendix

Definition 1 (Behavioural amalgamation)

Given the pushout diagram

\[
\begin{array}{c}
\text{BSPEC}_1 \quad \xrightarrow{i_1} \quad \text{BSPEC}_2 \\
\downarrow h_1 \quad \downarrow h_2 \\
\text{BSPEC}_3 \quad \xrightarrow{i_2} \quad \text{BSPEC}_4
\end{array}
\]

where \(i_1\) and \(h_1\) are \(\sigma\)-specie morhisms such that the pushout satisfies the observation preserving property, we define:

1. For all algebras \(A_3 \in \text{Beh}(\text{BSPEC}_3)\), \(A_2 \in \text{Beh}(\text{BSPEC}_2)\) and \(A_1 \in \text{Beh}(\text{BSPEC}_1)\) such that

\[
BU_{h_1}(A_3) = A_1 = BU_{i_1}(A_2)
\]

the behavioral amalgamated sum, or short b-amalgamation, of \(A_3\) and \(A_2\) with respect to \(A_1\), denoted by \(A_4 = A_3 \oplus_{A_1} A_2\), is the algebra in \(\text{Beh}(\text{BSPEC}_4)\) defined as

\[
A_4 = A_3 +_{A_1} A_2
\]

where \(+_{A_1}\) denotes the usual amalgamation in categories of algebras.

2. For all homomorphisms \(h_3 : A_3 \rightarrow B_3\) in \(\text{Beh}(\text{BSPEC}_3)\), \(h_2 : A_2 \rightarrow B_2\) in \(\text{Beh}(\text{BSPEC}_2)\) and \(h_1 : A_1 \rightarrow B_1\) in \(\text{Beh}(\text{BSPEC}_1)\) with

\[
BU_{h_1}(f_3) = f_1 = BU_{i_1}(f_2)
\]

the behavioral amalgamated sum, or short b-amalgamation, of \(f_3\) and \(f_2\) with respect to \(f_1\), denoted by \(f_4 = f_3 \oplus_{f_1} f_2\), is the \(\text{BSPEC}_4\)-behaviour morphism defined for every \(s\) in \(S_4\) as

\[
f_{4_s} = \begin{cases} f_3 & \text{if } s \in i_3(S_3) \\ f_2 & \text{else} \end{cases}
\]

where \(+_{f_1}\) denotes, as above, the usual amalgamation in categories of algebras.

Lemma 2 (Behavioural Amalgamation Lemma)

Given a pushout diagram as above b-amalgamation has the following properties:

1. Given algebras \(A_3 \in \text{Beh}(\text{BSPEC}_3)\), \(A_2 \in \text{Beh}(\text{BSPEC}_2)\) and \(A_1 \in \text{Beh}(\text{BSPEC}_1)\) such that
the b-amalgamation $A_4 = A_3 \oplus_{A_1} A_2$ is the unique algebra in Beh(BSPEC4), satisfying $A_3 = U_{i2}(A_4)$ and $A_2 = U_{h2}(A_4)$

Conversely, every $A_4 \in$ Beh(BSPEC4) has a unique representation $A_4 = A_3 \oplus_{A_1} A_2$ where $A_3 = BU_{i2}(A_4)$, $A_2 = BU_{h2}(A_4)$ and $A_1 = BU_{h1}(A_3) = BU_{i1}(A_2)$.

2 Given homomorphisms $h_3 : A_3 \rightarrow B_3$ in Beh(BSPEC3), $h_2 : A_2 \rightarrow B_2$ in Beh(BSPEC2) and $h_1 : A_1 \rightarrow B_1$ in Beh(BSPEC1) with

$$U_{h1}(f_3) = f_1 = U_{i1}(f_2)$$

the b-amalgamation $f_4 = f_3 \oplus f_1 f_2$ is the unique homomorphism satisfying $f_3 = BU_{i2}(f_4)$ and $f_2 = BU_{h2}(f_4)$.

Conversely, every BSPEC4-behaviour morphism $f_4 : A_4 \rightarrow B_4$ has a unique representation $f_4 = f_3 \oplus f_1 f_2$ where $f_3 = BU_{i2}(f_4)$, $f_2 = BU_{h2}(f_4)$ and $f_1 = BU_{h1}(f_3) = BU_{i1}(f_2)$.

**Proof idea**

1. The key to this proof is showing that if if the pushout satisfies the observation preserving property this means that:

$$\begin{array}{c}
\text{BSPEC1}^* \\
h_1 \\
\text{BSPEC3}^*
\end{array} \quad \begin{array}{c}
i_1 \\
\rightarrow \\
i_2
\end{array} \quad \begin{array}{c}
\text{BSPEC2}^* \\
h_2 \\
\text{BSPEC4}^*
\end{array}$$

is also a pushout diagram. This is enough to show that for the given $A_1$, $A_2$ and $A_3$, $A_4$ belongs to $\text{Alg(BSPEC4}^*)$ and, therefore, to Beh(BSPEC4). The rest of the proof of this part is just straightforward.

2. In this part, again, the key point is showing that $f_4 \in$ Beh(BSPEC4). This is done by induction on the structure of the $\Sigma$-observable computations and making use of the observation preserving property. []

**Lemma 3 (Behaviour Extension Lemma)**

1. Given a pushout diagram as above and a strongly persistent functor $F : \text{Beh}(\text{BSPEC1}) \rightarrow \text{Beh}(\text{BSPEC2})$ there is a persistent functor $F' : \text{Beh}(\text{BSPEC3}) \rightarrow \text{Beh}(\text{BSPEC4})$ called extension of $F$ via $h$ and written $F' = \text{EXT}(F,h)$ which is defined by the following b-amalgamations:

- $F'(A_3) = A_3 \oplus_{A_1} F(A_1)$ for every $A_3$ in Beh(BSPEC3), where $A_1 = BU_{h1}(A_3)$
- $F'(f_3) = f_3 \oplus f_1 F(f_1)$ for every $f_3$ in Beh(BSPEC3), where $f_1 = BU_{h1}(f_3)$

then, $F'$ is uniquely defined (up to isomorphism) by $BU_{h2}'F' = F'BU_{h1}$.

2. If $F$ is a free functor with respect to $BU_{i1}$ then $F'$ is free w.r.t. $BU_{i2}$.

**Proof idea**

1. As usual, this part is a direct consequence of the (behaviour) amalgamation lemma.
2. Suppose $B \in \text{Beh}(\text{BSPEC}3)$ and $f : A_3 \to B \cup_{i_2}(B_4) \in \text{Beh}(\text{BSPEC}3)$, then let $A_1 = B \cup_{h_1}(A_3)$, $B_2 = B \cup_{h_2}(B_4)$ and $f^* = B \cup_{h_1}(f)$. Since $F_{i_1}$ is free then there is a unique $g' : F(A_1) \to B$ in $\text{Beh}(\text{BSPEC}2)$ such that $B \cup_{i_1}(g') = f$. Taking $g = f \oplus_f g$ we have that $g : F(A_3) \to B_4$ and $B \cup_{i_2}(g) = f$. Moreover, $b$-amalgamation lemma implies the uniqueness of $g$.[1]

**Lemma 4 (View Extension Lemma)**

Given the pushout diagram:

\[
\begin{array}{c}
\text{BSPEC}1 & \xrightarrow{i_1} & \text{BSPEC}2 \\
h_1 & & \searrow h_2 \\
\downarrow & & \downarrow \\
\text{BSPEC}3 & \xrightarrow{i_2} & \text{BSPEC}4 \\
\end{array}
\]

where $h_1$ (and $h_2$) are view behaviour specification morphisms, $i_1$ and $i_2$ are behaviour specification morphisms and the pushout satisfies the observation preserving property, we have that:

- If $B\text{Free}_{i_1}$ is persistent then $B\text{Free}_{i_2}$ is also persistent.
- Moreover, $B\text{Free}_{i_2}$ is an extension of $i_1$ in the following sense

\[B\text{Free}_{i_1} \circ \text{View}_{h_1} = \text{Sig}_2 \circ \text{View}_{h_2} \circ B\text{Free}_{i_2}\]

**Proof idea**

The key point in this proof is the decomposition of the given diagram into

\[
\begin{array}{c}
\text{BSPEC}1 & \xrightarrow{i_1} & \text{BSPEC}2 \\
\downarrow & & \downarrow \\
\text{BSPEC}3 & \xrightarrow{i_2} & \text{BSPEC}4 \\
\end{array}
\]

where $\text{BSPEC}3^+ = (\text{Obs}3 + h_1(\text{Obs}1), \text{SP}3)$ and $\text{BSPEC}4^+ = (\text{Obs}4 + h_2(\text{Obs}2), \text{SP}4)$, $h_1^+$ and $h_2^+$ are defined (on sorts and operations) as $h_1$ and $h_2$, and $i_3$ and $i_4$ are the identity (on sorts and operations) morphisms. Then, persistency of $B\text{Free}_{i_1}$ implies persistency of $B\text{Free}_{i_2}$. Also, persistency of $B\text{Free}_{i_2}^+$ implies persistency of $B\text{Free}_{i_2}$ since the former can be seen as a realization of the latter.

On the other hand, again by the Behaviour Extension Lemma, $B\text{Free}_{i_2}^+$ is an extension of $B\text{Free}_{i_1}$. To prove

\[B\text{Free}_{i_2}^+ \circ \text{View}_{i_3} = \text{Sig}_2 \circ \text{View}_{i_4} \circ B\text{Free}_{i_2}\]

it is enough to take into account that the view functors are, in this case, free functors associated to the inverse morphisms $i_3^-$ and $i_4^-$ which are observable sort preserving although not necessarily non-observable sort preserving (cf. remark 2.1.2.2). Then the composition of free functors is also free and, therefore, $B\text{Free}_{i_2}^+ \circ \text{View}_{i_3}$ and $\text{View}_{i_4} \circ B\text{Free}_{i_2}$ are naturally isomorphic, which implies
It must be noted that for this kind of diagrams amalgamation cannot be defined in a way that satisfies the amalgamation lemma. The problem consists in that if we have a view behaviour specification morphism \( h: \text{BSPEC}_1 \to \text{BSPEC}_2 \) then, for a given \( A_2 \) in \( \text{BSPEC}_2 \), \( \text{View}_h(A_2) \) may be not behaviourally equivalent to \( \text{BU}_h(A_2) \). This happens, for instance, if

\[
\begin{align*}
\text{bspec BSPEC}_1 &= \text{sorts obs } s_1 \\
\text{ops } a, b : \to s_1
\end{align*}
\]

\[
\begin{align*}
\text{bspec BSPEC}_2 &= \text{sorts non obs } s_1 \\
\text{opns } a, b : \to s_1
\end{align*}
\]

\( h \) is the obvious inclusion, and \( A_2 \) is the algebra \( A_2 s_1 = \{v\} \), \( a_{A_2} = b_{A_2} = v \), then \( \text{BU}_h(A_2 s_1) = \{v\} \)
while \( \text{View}_h(A_2) = \{a, b\} \).

This means that amalgamation, satisfying the lemma, cannot exist since any reasonable definition of

\[
A_4 = A_3 \oplus_{A_1} A_2
\]

in the diagram above, should satisfy \( \text{BU}_h(A_4) = A_2 \) and, on the other hand, an adequate version of the amalgamation lemma would ask for \( \text{View}_h(A_4) =_{\text{Sig}_2} A_2 \).