THE SEMANTICS OF SHARED SUBMODULES SPECIFICATIONS*

E.K. Blum and F. Parisi-Presicce
Department of Mathematics
University of Southern California
Los Angeles, CA 90089-1113

ABSTRACT. After reviewing the concept of module specification with import and export interfaces introduced by H. Ehrig for the modular development of software systems, precise definitions of submodule and union of modules specifications are given along with some basic results on their compatibility and semantics. The notion of amalgamated sum is used for the semantics of unions of modules and some connections are made with parametrized specifications. The results are restricted to the basic algebraic case.

1. INTRODUCTION

In [2], a new algebraic specification concept, called a "module", was introduced for the modularization of software systems. A module is an abstract data type equipped with an import interface and an export interface: the import interface represents the operations available (and previously specified) inside the module while the export interface consists of the operations available to the user of the module. The two interfaces are combined in the body of the module, whose operations not in the export interface are considered "hidden". The interfaces are allowed to share a common parameter part.

The focus in [2] is on operations of composition of modules and actualization of parameterized modules and their interaction. It is obvious that these operations do not suffice for the construction of more complicated modules from simpler ones and that , in practical languages such as Ada, the union operation is an important one. The mere formation of the union of two

*This research was supported in part by the National Science Foundation under Grant MCS 82-03666.

disjoint modules poses no algebraic difficulties, nor does the formation of the union of modules which share a common part, provided we are willing to duplicate this common part. In many situations, however, having two copies of a common part leads to difficulties in using the union module in a wider context requiring importing or exporting the common part. Therefore, we need a kind of union which allows us to avoid such duplication.

In this paper, we describe, in increasing order of generality, the union of modules which share the parameter part, a subparameter part and finally a complete submodule. This last situation requires the precise definition of submodule specification and its semantics, which are given in Section 4. The results, in Section 5, relating the semantics of the union module to those of its components are described in terms of amalgamated sums of algebras. This notion is introduced in Section 2, where we present a constructive definition of the amalgamated sum $A1 +_{A0} A2$ of two algebras A1 and A2 with respect to a third algebra A0. The amalgamated sum is then related to pullback constructs in the category of SPEC-algebra categories and to pushouts in a new category <u>UAlg</u>. In Section 3, we briefly illustrate the connection between amalgamated sums and loose semantics of standard parameter passing in parameterized data types. Further developments pointing the way to an algebra of modules are outlined in Section 6.

2. AMALGAMATED SUM

In this section, we introduce the concept of <u>amalgamated sum</u> and relate its constructive definition to other well known constructions in the framework of category theory. Let SPECi, i = 0,1,2,3, be algebraic specifications and **SPECi** the corresponding categories of SPECi-algebras. Also let f_i: SPEC0 → SPECi and g_i: SPECi → SPEC3, for i = 1,2, be specification morphisms and Ui and Vi the forgetful functors associated with the specification morphisms g_i and f_i respectively, such that (1) is a pushout diagram in the category of algebraic specifications and (2) is a pullback diagram in the category of SPEC-algebras categories and forgetful functors.

$$\begin{array}{ccc} \mathrm{SPEC0} & \xrightarrow{f1} & \mathrm{SPEC1} \\ {\scriptstyle f2}\downarrow & (1) & \downarrow{\scriptstyle g1} \\ \mathrm{SPEC2} & \xrightarrow[g2]{} & \mathrm{SPEC3} \end{array} \quad . \quad \begin{array}{ccc} \mathbf{SPEC0} & \xleftarrow{V1} & \mathbf{SPEC1} \\ {\scriptstyle V2}\uparrow & (2) & \uparrow{\scriptstyle U1} \\ \mathbf{SPEC2} & \xleftarrow[U2]{} & \mathbf{SPEC3} \end{array}$$

2.1 Definition (Amalgamated Sum)

Given algebras $Ai \in$ **SPECi** for $i = 0,1,2$ with the property that $V1(A1) = A0 = V2(A2)$, the amalgamated sum $A1+_{A0}A2$ of A1 and A2 with respect to A0 is the unique SPEC3-algebra A3 satisfying the following conditions on sorts $s \in S3$ and operators $\sigma \in \Sigma3$

$$(A3)_s = \begin{cases} A1_{s1} \text{ if } g1(s1) = s \\ A2_{s2} \text{ if } g2(s2) = s \end{cases}$$

$$\sigma_{A3} = \begin{cases} \sigma1_{A1} \text{ if } g1(\sigma1) = \sigma \\ \sigma2_{A2} \text{ if } g2(\sigma2) = \sigma \end{cases} .$$

The algebra A3 is well-defined, since SPEC3 is the pushout of SPEC1 and SPEC2. Thus if $s \in S3$, then $g1(s1) = s$ for some $s1 \in S1$ or $g2(s2) = s$ for some $s2 \in S2$ and if both are true, then there exists $s0 \in S0$ such that $fi(s0) = si$, $i = 1,2$ in which case $(A1)_{s1} = (A0)_{s0} = (A2)_{s2}$. A similar argument shows that σ_{A3} is also well defined and that, in fact, $A1+_{A0}A2$ is a SPEC3-algebra.

The amalgamated sum $A1+_{A0}A2$ can also be defined implicitly in terms of the pullback diagram of the SPECi-algebra categories.

2.2 Lemma (Pullback Property)

Given $Ai \in$ **SPECi**, $i = 0,1,2$, with $V1(A1) = A0 = V2(A2)$, the amalgamated sum $A3 = A1+_{A0}A2$ is the unique SPEC3-algebra such that $U1(A3) = A1$, $U2(A3) = A2$ and if $B \in$ **SPEC** for an algebraic specification SPEC with (forgetful) functors Fi: **SPEC** → **SPECi**, $i = 1,2$, $V1 \cdot F1 = V2 \cdot F2$ and $Fi(B) = Ai$, then there exists a unique (forgetful) functor F: **SPEC** → **SPEC3** such that $Fi = Ui \cdot F$, $i = 1,2$, and $F(B) = A3$.

2.3 Corollary

$$\textbf{SPEC3} = \textbf{SPEC1} +_{\textbf{SPEC0}} \textbf{SPEC2} = \{A1 +_{A0} A2 : \; Ai \in \textbf{SPECi},\; V1(A1) = A0 = V2(A2)\}$$

The amalgamated sum $A1 +_{A0} A2$ can also be viewed as the pushout of A1 and A2 w.r.t. A0 in the appropriate category. In order to define this category, we need the notion of "generalized homomorphism".

2.4 Definition (Generalized Homomorphism)

Let SPECi be algebraic specifications and Ai be SPECi-algebras for i = 0, 1. A generalized homomorphism from (A0, SPEC0) to (A1, SPEC1), denoted by (h,f): (A0, SPEC0) $\to$ (A1, SPEC1) is a pair of functions (h,f) where f: SPEC0 $\to$ SPEC1 is a specification morphism and h is a family h_s: $A0_s \to A1_{f(s)}$ of functions indexed by the set S0 of sorts in SPEC0 such that, for every σ: s1x ... xsn $\to$ s in $\Sigma 0$, the following diagram commutes:

$$\begin{array}{ccc} A0_{s1} \times \cdots \times A0_{sn} & \xrightarrow{\sigma_{A0}} & A0_s \\ {\scriptstyle h_{s1} \times \cdots \times h_{sn}} \downarrow & & \downarrow {\scriptstyle h_s} \\ A1_{f(s1)} \times \cdots \times A1_{f(sn)} & \xrightarrow{f(\sigma)_{A1}} & A1_{f(s)} \end{array}$$

Given f: SPEC0 $\to$ SPEC1, we denote by $GENHOM_f(A0,A1)$ the set of all functions h such that (h,f): (A0, SPEC0) $\to$ (A1, SPEC1) is a generalized homomorphism.

Notice that any h $\in GENHOM_f(A0,A1)$ is also a SPEC0-morphism from A0 to $V_f(A1)$ since h_s: $A0_s \to A1_{f(s)} = V_f(A1)_s$ and $\sigma_{V_f(A1)}$ is defined as $f(\sigma)_{A1}$. Conversely, if h $\in$ **SPEC0**$(A0,V_f(A1))$, then h_s: $A0_s \to V_f(A1)_s = A1_{f(s)}$ and the above diagram commutes since V_f is a functor. We have just proved the first part of the following result.

2.5 Proposition

$GENHOM_f(A0,A1) \cong$ **SPEC0**$(A0,V_f(A1)) \cong$ **SPEC1**$(F_f(A0),A1)$.

The second isomorphism follows from general properties of the forgetful functor V_f and its left adjoint, the free functor F_f.

2.6 Definition

Given generalized homomorphisms (h0,f0): (A0,SPEC0) $\to$ (A1,SPEC1) and (h1,f1): (A1,SPEC1) $\to$ (A2,SPEC2), the composition is given by the pair (h1 $\bullet$ h0, f1 $\bullet$ f0) and it is clear, either directly from Definition 2.4 or using Proposition 2.5, that it is again a generalized homomorphism. It is also clear that the composition of generalized homomorphisms, when defined, is associative. We denote by UAlg the category with objects the pairs (A, SPEC) with A $\in$ **SPEC** and morphisms the generalized morphisms (h,f): (A0,SPEC0) $\to$ (A1,SPEC1). We can now state the main result of this section.

2.7 Proposition

Given specification morphisms fi: SPEC0 $\to$ SPECi and SPECi-algebras Ai, i = 0, 1, 2, satisfying V1(A1) = A0 = V2(A2), the amalgamated sum $A1+_{A0}A2$ is the pushout of A1 and A2 w.r.t. A0, that is, the diagram

$$\begin{array}{ccc} (A0,SPEC0) & \xrightarrow{(h1,f1)} & (A1,SPEC1) \\ {\scriptstyle (h2,f2)}\downarrow & & \downarrow{\scriptstyle (k1,g1)} \\ (A2,SPEC2) & \xrightarrow[(k2,g2)]{} & (A1+_{A0}A2,SPEC3) \end{array}$$

is a pushout diagram in the category UAlg, where hi and ki are the obvious inclusion maps satisfying $hi_s(A0_s) = Ai_{fi(s)}$ and $ki_{si}(Ai) = (A1+_{A0}A2)_{gi(si)}$.

Remark Notice that if SPEC0 = ϕ in diagram (1), then SPEC3 is the disjoint union of SPEC1 and SPEC2 and every algebra A3 $\in$ **SPEC3** is the disjoint union of an algebra A1 $\in$ **SPEC1** and an algebra A2 $\in$ **SPEC2**.

3. PARAMETERIZED DATA TYPES AND AMALGAMATED SUMS

The notion of parameterized data type is an important one in the hierarchical design of large programming systems. While several authors ([1,4,6]) have considered the problems of abstract data type specifications and implementations, the problem of parameter passing has not received as much attention ([3,5]). Here, we look at amalgamated sums of algebras as a "constructive" parameter passing technique. We take the following definition from [5].

3.1 Definition (Parameterized Data Type)

A parameterized data type PDT = (SPEC0, SPEC1, T) consists of two algebraic specifications SPEC0 and SPEC1 with SPEC0 $\subseteq$ SPEC1 (componentwise) and a functor T: **SPEC0** $\to$ **SPEC1** which is assumed to be strongly persistent, i.e. V(T(A)) = A for every A $\in$ **SPEC0**, where V is the forgetful functor associated with the inclusion specification morphism j: SPEC0 $\to$ SPEC1.

In the case of initial algebra semantics [5,6], the functor T is taken to be the free functor F: **SPEC0** $\to$ **SPEC1**. In order to pass an actual parameter specification SPEC2 for the parameter part SPEC0 of a parameterized specification PSPEC = (SPEC0,SPEC1), a "parameter passing" morphism h: SPEC0 $\to$ SPEC2 is specified and a new specification SPEC3 is constructed as in the following pushout diagram:

$$\begin{array}{ccc} SPEC0 & \xrightarrow{j} & SPEC1 \\ {\scriptstyle h}\downarrow & & \downarrow{\scriptstyle h'} \\ SPEC2 & \xrightarrow{j'} & SPEC3 \end{array}$$

(In [5], the construction of the new specification is explicit). The semantics of this standard (i.e. non-parameterized) parameter passing is taken to be $(F, T_{SPEC2}, T_{SPEC3})$, where T_{SPECi} is the initial algebra in **SPECi** and F: **SPEC0** $\to$ **SPEC1** is the free functor of the parameterized specification PSPEC. The assumption that F be strongly persistent is then sufficient to guarantee the semantical conditions:

1) actual parameter protection: $V_{j'}(T_{SPEC3}) = T_{SPEC2}$
2) passing compatibility: $F(V_h(T_{SPEC2})) = V_{h'}(T_{SPEC3})$.

In the loose semantics case, the result of passing a SPEC2-algebra as actual parameter can be expressed as an amalgamated sum of algebras. Let again PSPEC, SPEC2, h: SPEC0 $\to$ SPEC2 and SPEC3 be given as above. Let $A2 \in$ **SPEC2** and define $A0 = V_h(A2)$. Then $A1 = T(A0)$ is a SPEC1-algebra with the property that $V(A1) = A0$ (by strong persistency of T). Since A0, A1 and A2 satisfy the assumptions of Definition 2.1, we can define their amalgamated sum $A3 = A1 +_{A0} A2$. Then $A3 = T(V_h(A2)) +_{V_h(A2)} A2$ is the result of passing A2 to the parameterized data type PDT = (SPEC0,SPEC1,T). It is easy now to check that by the properties of the amalgamated sum, similar semantical conditions are satisfied. The actual parameter A2 is protected since

$$V_{j'}(A3) = V_{j'}(T(A0) +_{A0} A2) = A2$$

and the parameter passing in "compatible", i.e. it reflects the behavior of the functor T, since $T(V_h(A2)) = T(A0) = A1 = V_{h'}(A3)$. Hence **SPEC1** $+_{\mathbf{SPEC0}}$ **SPEC2** (see Corollary 2.3) can be taken as the loose semantics of (standard parameter) passing SPEC2 for SPEC0 in PDT.

4. MODULE AND SUBMODULE SPECIFICATIONS

In this section, we first review the basic notions of module specification with import and export interfaces as introduced by Ehrig ([2]) briefly mentioning the operations of composition and actualization and their semantics. We then introduce the notion of submodule specification and semantics to be used in the next section in the context of unions of modules sharing a common part.

4.1 Definition (Module Specification)

A module specification MOD consists of four algebraic specifications PAR, IMP, EXP, BOD along with specification morphisms e, s, i and v (e and s injective) making the following diagram commute:

$$\begin{array}{ccc} PAR & \xrightarrow{\ e\ } & EXP \\ {\scriptstyle i}\downarrow & & \downarrow{\scriptstyle v} \\ IMP & \xrightarrow{\ s\ } & BOD \end{array}$$

IMP and EXP are the import and export interfaces, respectively, and PAR is the parameter part shared by IMP and EXP. We will assume that e and s are actually inclusions.

4.2 Definition (Semantics of Modules)

Given a module specification MOD as in Definition 4.1, denote by V_s, V_v and V_e the forgetful functors induced by s, v and e, respectively, and by FREE: **IMP** $\rightarrow$ **BOD** the free functor associated with V_s.

The <u>(unrestricted) semantics</u> SEM of MOD is the functor

$$SEM = V_v \cdot FREE\colon \mathbf{IMP} \rightarrow \mathbf{EXP}.$$

The <u>restriction semantics</u> RSEM of MOD is the functor

$$RSEM = R \cdot SEM\colon \mathbf{IMP} \rightarrow \mathbf{EXP}$$

where, for $A \in \mathbf{EXP}$, $R(A) = \bigcap\{B \in \mathbf{EXP}\colon\ B \subseteq A,\ V_e(B) = V_e(A)\}$.

<u>Assumptions</u> Using the unrestricted semantics SEM, we will assume that FREE is strongly persistent (i.e. $V_s \cdot$ FREE is the identity on **IMP**). Using RSEM, we will add the assumption that FREE preserves injective homomorphisms. For a discussion of the interpretation of both definitions, see [2].

In the composition of two modules the import interface of one module is "matched" with the export interface of the other one.

4.3 Definition (Composition of Modules)

Given two modules specifications MODi = (PARi, EXPi, IMPi, BODi) with a specification morphism h: IMP1 $\rightarrow$ EXP2, the composition of MOD1 and MOD2 w.r.t. h, denoted by MOD2 $\cdot_h$ MOD1, is the module specification MOD3 = (PAR3, EXP1, IMP2, BOD3) with PAR3 and BOD3 defined as in the diagram

$$\begin{array}{ccccc} PAR3 & \longrightarrow & PAR1 & \longrightarrow & EXP1 \\ \downarrow & (1) & \downarrow & & \downarrow \\ & & IMP1 & \longrightarrow & BOD1 \\ & & \downarrow & & \\ PAR2 & \longrightarrow & EXP2 & (2) & \downarrow \\ \downarrow & & \downarrow & & \\ IMP2 & \longrightarrow & BOD2 & \longrightarrow & BOD3 \end{array}$$

where (1) and (2) are a pullback and a pushout diagram respectively.

4.4 Theorem (Semantics of Composition)

i) $SEM3 = SEM1 \cdot V_h \cdot SEM2$

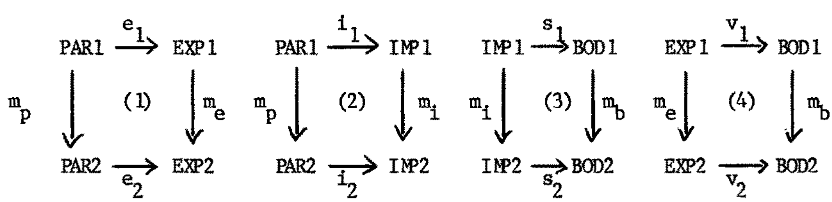
ii) If h: IMP1 → EXP2 is "parameter consistent", i.e. there exists p: PAR1 → PAR2 such that $e_2 \cdot p = h \cdot i_1$, then $RSEM3 = RSEM1 \cdot V_h \cdot RSEM2$.

The other operation on modules mentioned in the introduction is that of actualization, where the parameter part PAR0 of a parametrized module MOD0 is replaced by a specification ACT (actual parameter) to yield a parameterless module specification. The actualization of MOD0 by ACT w.r.t a specification morphism h: PAR0 → ACT is the parameterless module $ACT_h(MOD0) = (\phi$, EXP, IMP, BOD) where EXP (resp. IMP) is the pushout of ACT and EXP0 (resp. IMP0) w.r.t. PAR0 and BOD is obtained by "gluing" IMP and BOD0. For the precise definition and results dealing with the induced semantics of actualization and compatibility properties of composition and actualization, see [2].

We now introduce the concept of submodule specification. As in [2], we restrict our attention to the basic algebraic case, without logical or algebraic constraints on the interfaces.

4.5 Definition (Submodule Specification)

Given two module specifications MODi = (PARi, EXPi, IMPi, BODi) for i = 1,2, MOD1 is a submodule specification of MOD2 if there exist four specification morphisms m_p: PAR1 → PAR2, m_e: EXP1 → EXP2, m_i: IMP1 → IMP2 and m_b: BOD1 → BOD2 such that the following four diagrams commute:



Assumptions We have already assumed that, in module specifications, the free functor FREEi: **IMPi** → **BODi** is strongly persistent (when using unrestricted semantics) or strongly conservative (with restriction semantics). In the case of submodule specification, we will add the condition that the free functors FREE1 and FREE2 commute with the vertical forgetful functors of diagram 3, i.e. $V_{m_b} \cdot FREE2 = FREE1 \cdot V_{m_i}$. This formalizes our intuitive notion that, for MOD1 to be a submodule of MOD2, the free construction in MOD1 should reflect the free construction in MOD2. When using the restriction semantics, we will also add the assumption that, for every EXP2-algebra A, $V_{m_e}(R2(A)) = R1(V_{m_e}(A))$. These assumptions are sufficient to relate the semantics of MOD1 and MOD2.

4.6 Proposition (Submodule Semantics)

Given module specifications MOD1 and MOD2 with MOD1 a submodule specification of MOD2 and the above assumptions on the behavior of the forgetful functors, we have

i) $V_{m_e} \cdot SEM2 = SEM1 \cdot V_{m_i}$

ii) $V_{m_e} \cdot RSEM2 = RSEM1 \cdot V_{m_i}$

where SEMi, RSEMi: **IMPi** $\rightarrow$ **EXPi**, V_{m_e} : **EXP2** $\rightarrow$ **EXP1** and V_{m_i} : **IMP2** $\rightarrow$ **IMP1**.

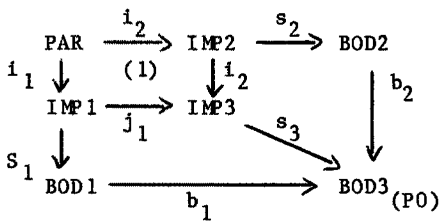
Remarks If in Definition 4.5 we take PAR1 = PAR2, then the notion of MOD1 being a submodule specification of MOD2 is equivalent to MOD2 being a "refinement" of MOD1 with the additional specification morphism m_b: BOD1 $\rightarrow$ BOD2 (see [2] sec. 4.5). In view of Proposition 4.6, our assumptions on submodule specifications imply Ehrig's notions of "correct" and "R-correct" refinements. If in addition we take IMP1 = IMP2 and diagram (4) as a pushout, then we obtain a special case of an "extension" of module specification as in 4.6 of [2].

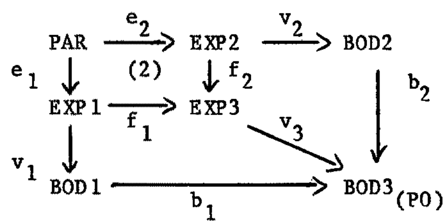
5. UNION OF MODULES WITH SHARED SUBMODULES

As mentioned already in the Introduction, composition and actualization are but two of the operations that can be used to build up complex modules from simpler ones. Another possible construction, allowed, for example, in Ada, is that of a union of two (or more) modules. A larger module can be obtained whose import and export interfaces are formed by combining the import and export interfaces of the component modules, respectively. The simplest possible combination is that of a union of two disjoint modules or, equivalently, of two modules which share a common part, may it be a submodule or just part of an interface, and we are willing to duplicate that part in the composite module. There are instances, however, where two modules share a common part, say the parameter part, which should not be duplicated since PAR is intended to be instantiated, at a later stage in the development, with the same actual parameter. This is the situation we analyze next.

5.1 Definition (Union of Modules with Shared Parameter)

The union of two modules specifications MODi = (PAR, EXPi, IMPi, BODi) for i = 1,2, which share the parameter part PAR, is denoted by $MOD1 +_{PAR} MOD2$ and is the module specification MOD3 = (PAR, EXP3, IMP3, BOD3) where the last three specifications are given by the pushout diagrams:



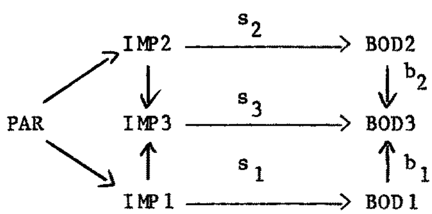


By definition of MOD1 and MOD2, the two outer diagrams are the same and they define BOD3 as a pushout w.r.t. PAR. Since (1) and (2) are pushouts, s_3 and v_3 exist and are unique. They are also injective and $v_3 \cdot e_3 = b_2 \cdot v_2 \cdot e_2 = b_2 \cdot s_2 \cdot i_2 = s_3 \cdot i_3$.

The following Lemma is needed to prove Theorem 5.3.

5.2 Lemma

Given the diagram



with IMP3 and BOD3 as in Definition 5.1, let Vi be the forgetful functor associated with si and Fi be the corresponding free functor. Define $F = F1 +_{PAR} F2$: **IMP3** → **BOD3** by letting $F(I1 +_P I2) = F1(I1) +_P F2(I2)$
Then F = FREE3 is the free functor associated with s_3 and if F1 and F2 are strongly persistent (resp. conservative), then so is F.

5.3 Theorem (Semantics of Union with Shared Parameter)

Given the module specification $MOD3 = MOD1 +_{PAR} MOD2$ as in Definition 5.1, its semantics are given by

i) $SEM3 = SEM1 +_{PAR} SEM2$

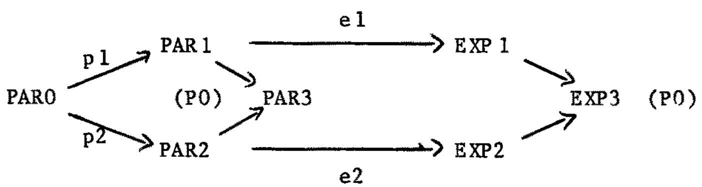
ii) $RSEM3 = RSEM1 +_{PAR} RSEM2$

where $(SEM1 +_{PAR} SEM2)(I1 +_P I2) = SEM1(I1) +_P SEM2(I2)$
and $RSEM1 +_{PAR} RSEM2$ is defined similarly.

The next situation we consider is that of a union of two module specifications MODi = (PARi, EXPi, IMPi, BODi) for i = 1,2 where PAR1 and PAR2 share a common subparameter part PAR0 which should not be duplicated in the union. The sharing of this common subparameter is indicated by two specification morphisms pi: PAR0 → PARi for i = 1,2.

5.4 Definition (Union of Modules with Shared Subparameter)

Given module specifications MODi = (PARi, EXPi, IMPi, BODi) for i = 1,2 and a specification PAR0 with specification morphisms pi: PAR0 → PARi, the union MOD1 $+_{PAR0}$MOD2 of MOD1 and MOD2 w.r.t. PAR0 is the module specification MOD3 = (PAR3, EXP3, IMP3, BOD3) where PAR3 is defined as the pushout of PAR1 and PAR2 w.r.t. PAR0, and EXP3, IMP3 and MOD3 are obtained as in Definition 5.1 with PAR0 replacing PAR, e.g.



(3)

If PAR0 = PAR1 = PAR2, we are back to the case of Shared Parameter, while if PAR0 = ϕ we have disjoint union. The same arguments can be used to show that the diagram of MOD3 commutes and that FREE3: **IMP3** → **BOD3** is nothing more than FREE1 $+_{PAR0}$FREE2 and is again strongly persistent (conservative) whenever FREE1 and FREE2 are.

5.5 Theorem (Semantics of Union with Shared Subparameters)

The unrestricted and restriction semantics SEM3 and RSEM3, respectively, of MOD3 = MOD1 $+_{PAR0}$MOD2 as in Definition 5.4 are given by

i) SEM3 = SEM1 $+_{PAR0}$SEM2 and

ii) RSEM3 = RSEM1 $+_{PAR0}$RSEM2.

The only situations considered so for are those involving union of either disjoint modules or module sharing part or all of the parameter part. In the more general situation, two modules to be combined can share part (or all) of the import and/or export interfaces, and therefore part (or all) of the body.

5.6 Definition (Union of Modules with Shared Submodule)

Given a submodule MOD0 = (PAR0, EXP0, IMP0, BOD0) of two module specifications MODj = (PARj, EXPj, IMPj, BODj) for j = 1,2 with specification morphisms m_{pj}: PAR0 → PARj, m_{ej}: EXP0 → EXPj, m_{ij}: IMP0 → IMPj, m_{bj}: BOD0 → BODj for j = 1,2 as in Definition 4.5, the union of MOD1 and MOD2 with shared MOD0, denoted by MOD1 $+_{MOD0}$MOD2, is the module specification MOD3 = (PAR3, EXP3, IMP3, BOD3) where each of its specifications is given as a pushout of the corresponding specifications in MOD0, MOD1 and MOD2 with the appropriate specification morphisms.

Remark In this definition of union, the parts shared by MOD1 and MOD2 are required to form a submodule of both MOD1 and MOD2. According to our assumptions in Definition 4.5, this implies not only that FREE0: **IMP0** → **BOD0** is strongly persistent (or conservative) but also that $V_{m_{bj}} \cdot$ FREEj = FREE0 $\cdot V_{m_{ij}}$ for j = 1,2 and that $V_{m_{ej}} \cdot$ Rj = R0 $\cdot V_{m_{ej}}$ for j = 1,2.

If the two modules share only a subparameter, then we can take PAR0 = EXP0 = IMP0 = BOD0 and this union reduces to the one given in Definition 5.4. If only part of the export interface (and, therefore, of the body) is shared, we can take PAR0 = IMP0 = ϕ and EXP0 = BOD0, while if the shared part is in the import interface, we let PAR0 = EXP0 = ϕ but we still require the free functor from **IMP0** to **BOD0** to be strongly persistent (or conservative).

The semantics of the union of two modules with a shared submodule behaves exactly as we expect it (or hope for it) to behave.

5.7 Theorem (Semantics of Union with Shared Submodule)

The semantics SEM3 of the union module specification MOD3 = MOD1 $+_{MOD0}$MOD2 is the amalgamated sum of the semantics of MOD1 and MOD2 w.r.t. the semantics of MOD0, i.e. SEM3 is uniquely defined by SEM3 = SEM1 $+_{SEM0}$SEM2.

Proof Let Vj: **BODj** → **EXPj** denote the forgetful functor associated with the specification morphism vj: EXPj → BODj. We first prove that V3 = V1 $+_{V0}$V2, i.e. V3(B1 $+_{B0}$B2) = V1(B1) $+_{V0(B0)}$V2(B2), where Bj ∈ **BODj** for j = 0,1,2 and B1 $+_{B0}$B2 ∈ **BOD3**. Since MOD0 is a submodule of both MOD1 and MOD2, V0 $\cdot V_{m_{bj}} = V_{m_{ej}} \cdot$ Vj or, equivalently, V0(B0) = Vj(Bj)$_{EXP0}$ for j = 1,2. Then (V3(B1 $+_{B0}$B2))$_{EXPj}$ = ((B1 $+_{B0}$B2)$_{EXP3}$)$_{EXPj}$ = Bj$_{EXPj}$ = Vj(Bj) for j = 0,1,2 and therefore V3(B1 $+_{B0}$B2) = V1(B1) $+_{V0(B0)}$V2(B2) by uniqueness of the amalgamated sum (Lemma 2.2 or Proposition 2.7). We now show that, if we define F3: **IMP1** $+_{\mathbf{IMP0}}$**IMP2** → **BOD1** $+_{\mathbf{BOD0}}$**BOD2** by F3(I1 $+_{I0}$I2) = F1(I1) $+_{F0(I0)}$F2(I2), where Ij ∈ **IMPj** and Fj is the free functor from **IMPj** to **BODj** for j = 0,1,2, then F3 is the free functor from **IMP3** to **BOD3** and is strongly persistent if F0, F1 and F2 are. To this extent, let

$$f3 = f1 +_{f0} f2 \colon I1 +_{I0} I2 \to V_{s3}(B1 +_{B0} B2) = V_{s1}(B1) +_{V_{s0}(B0)} V_{s2}(B0)$$

be an IMP3-morphism. Since Fj is the free functor, there exists a unique BODj-morphism $\overline{f}j\colon Fj(Ij) \to Bj$ making the diagram

$$\begin{array}{ccc} Ij & \xrightarrow{fj} & V_{sj}(Bj) \\ {\scriptstyle \eta j}\downarrow & \nearrow & V_{sj}(\overline{f}j) \\ V_{sj}(Fj(Ij)) & & \end{array}$$

commute.

Then the BOD3-morphism $\overline{f1} +_{\overline{f0}} \overline{f2}$ makes the diagram

$$\begin{array}{ccc} I1 +_{I0} I2 & \xrightarrow{f3} & V_{s3}(B1 +_{B0} B2) \\ \downarrow & \nearrow & V_{s3}(\overline{f1} +_{\overline{f0}} \overline{f2}) \\ V_{s3}(F1(I1) +_{F0(I0)} F2(I2)) & & \end{array}$$

commute. Furthermore, F3 is strongly persistent since

$$V_{s3}(F1(I1) +_{F0(I0)} F2(I2)) = V_{s1}(F1(I1)) +_{V_{s0}(F0(I0))} V_{s2}(F2(I2))$$
$$= I1 +_{I0} I2 \text{ if F0, F1 and F2 are strongly persistent.}$$

Finally, $SEM3 = V3 \cdot F3 = (V1 +_{V0} V2) \cdot (F1 +_{F0} F2) = (V1 \cdot F1) +_{(V0 \cdot F0)} (V2 \cdot F2) =$
$= SEM1 +_{SEM0} SEM2$.

5.8 Theorem (Restriction Semantics)

The restriction semantics RSEM3 of $MOD3 = MOD1 +_{MOD0} MOD2$ is uniquely given by $RSEM3 = RSEM1 +_{RSEM0} RSEM2$.

Proof Since $RSEM3 = R3 \cdot SEM3$ the result will be established as soon as we show that $R3 = R1 +_{R0} R2$.

First notice that, for $Ej \in$ **EXPj**, $j = 0,1,2$, $R1(E1) +_{R0(E0)} R2(E2) \subseteq$ $E1 +_{E0} E2$ and that $(R1(E1) +_{R0(E0)} R2(E2))_{PAR3} = R1(E1)_{PAR1} +_{R0(E0)_{PAR0}} R2(E2)_{PAR2} =$
$= E1_{PAR1} +_{E0_{PAR0}} E2_{PAR2} = (E1 +_{E0} E2)_{PAR3}$ and hence $R3(E1 +_{E0} E2) \subseteq$
$R1(E1) +_{R0(E0)} R2(E2)$.

(Notice that we use here the assumption made in Definition 4.5 that

$V_{m_{ej}}(Rj(Ej)) = R0(V_{m_{e0}}(Ej)) = R0(E0)$ for $j = 1,2$).

On the other hand, if $R3(E1 +_{E0} E2) = \overline{E1} +_{\overline{E0}} \overline{E2}$ with $\overline{Ej} \subseteq Ej$, $\overline{Ej}_{PARj} = Ej_{PARj}$

and $R3(E1 +_{E0} E2)_{PARj} = \overline{Ej}$, then $Rj(Ej) \subseteq \overline{Ej}$ and therefore $R1(E1) +_{R0(E0)} R2(E2) \subseteq \overline{E1} +_{\overline{E0}} \overline{E2}$. Hence $R3(E1 +_{E0} E2) = R1(E1) +_{R0(E0)} R2(E2)$.

6. CONCLUSION AND FURTHER DEVELOPMENTS

Let us first give a short summary of the main constructions and results of this paper. In Section 4, after reviewing the basic concepts of module specification and semantics (as in [2]), we have introduced the concept of a submodule M of module M', imposed some (natural) restrictions on the connecting specification morphisms and related the semantics of M and M'. A precise notion of submodule is not only worthy of independent investigation, but also important for a precise treatment of the union of modules which share common parts. Different possible unions of modules have been presented (in Section 5) in increasing degree of difficulty, from the simple case of shared parameter to the most general one of union of modules with shared submodules. Both the unrestricted and the restriction semantics of the union modules have been shown to relate in a natural way to the semantics of its components. In discussing the semantics of the union of modules, we have made use of the notion of amalgamated sum of algebras, whose basic definition and properties have been introduced in Section 2. Connections between amalgamated sums and parametrized data types have been briefly touched upon in Section 3, where parameter passing has been formulated from a constructive point of view in both the initial and loose semantics cases.

Several questions arise from the developments in this paper and in [2] and are currently under investigation. Among the results that will be presented in full details in forthcoming papers, are some compatibility conditions on union and composition of modules that guarantee distributivity properties of these two operations, such as $(M1 + M2) \cdot M3 = (M1 \cdot M3) +_{M3\phi} (M2 \cdot M3)$ where $M3\phi$ is the submodule of M3 given by $M3\phi = (\phi, \phi, IMP3, BOD3)$. Similar results can also be obtained with unions of modules with shared submodules. Composition on the left, e.g. $M1 \cdot (M2 +_{M0} M3)$, seems to be more complicated, but we have some encouraging preliminary results of a "pseudo-distributive" nature. Results of this type are a prerequisite to a comprehensive development of an algebra of modules. The possibility of partial composition, i.e. the matching of only part of the import interface of a module with the export interface of another one, is also under investigation as a first step toward the construction of complex modules using a "recursive-like" interaction of simpler ones. The compatibility of the operations of union and actualization has been investigated in [7]. An example of shared submodules can be constructed from the example in [2]. This will be given in an expanded version of this paper.

Acknowledgements. The results in this paper are part of ongoing research being conducted jointly with H. Ehrig, Technische Universitat Berlin, and initiated during his visit to the University of Southern California in the Spring 1984. A more comprehensive account of this research will be presented in a forthcoming joint paper. Many thanks to Cynthia Summerville for fast and accurate typing.

References

[1] Blum, E.K., Parisi-Presicce, F., Implementation of Data Types by Algebraic Methods, J. Comput. System Sci. 27, 2 (Oct. 1983) 304-330.

[2] Ehrig, H., An Algebraic Specification Concept for Modules, Draft Version, Techn. Report No. 84-02, TU Berlin, FB 20, March 1984.

[3] Ehrig, H., Kreowski, H.-J., Compatibility of Parameter Passing and Implementation of Parameterized Data Types, Theoret. Comp. Sci. 27(1983) 255-286.

[4] Ehrig, H., Kreowski, H.-J., Mahr, B., Padawitz, P., Algebraic Implementation of Abstract Data Types, Theoret. Comp. Sci. 20(1982) 209-264.

[5] Ehrig, H., Kreowski, H.-J., Thatcher, J.W., Wagner, E.G., Wright, J.B., Parameter Passing in Algebraic Specification Languages, Proc. Aarhus Workshop on Prog. Spec., 1981, LNCS 134(1982) 322-369.

[6] Goguen, J.A., Thatcher, J.W., Wagner, E.G., An Initial Algebra Approach to the Specification, Correctness and Implementation of Abstract Data Types, Current Trends in Prog. Method., IV: Data Structuring (R.T. Yeh, Ed.) Prentice-Hall, NJ (1978) 80-149.

[7] Parisi-Presicce, F., The operations of union and actualization of module specifications are compatible, Extended Abstract, Univ. of Southern California, September 1984.