• 1400008489
còpia 1

Developement of algebraic specifications with constraints

F. Orejas
V. Sacristán
S. Clerici

Report LSI–89–10

Abstract: A formal framework for the development of algebraic specifications is presented. Ther main issues concerning the approach are the following: We allow to deal with incomplete specifications during the design process. This is handled by means of loose semantics and initial constraints. The design process is considered bi-dimensional. Horizontal refinements express, as usual, extensions. Vertical refinements consists in adding more detail or completing the refined specifications. The usual composition properties for refinements hold in our framework. In addition, the horizontal composition theorem defines a generalization of parameter passing as it is usually understood.

Development of Algebraic Specifications with Constraints

F. Orejas, V. Sacristán, S. Clerici
Facultat d'Informàtica, Universitat Politècnica de Catalunya
Pau Gargallo 5, (08028) Barcelona, SPAIN

Program design may be considered as a mixture of two activities. On one hand, programming consists on determining precisely the problem to be solved and, on the other, in describing a (hopefully) efficient solution on a given programming language.

Traditionally, these two activities have been more or less mixed. In classical waterfall models of software design, coding was the phase in which the problem to be solved was stated without ambiguity for the first time. In previous phases (design or requirements analysis), the problem was studied and more or less clarified but, in general, never formally stated. Then, the final code was, at the same time, the problem statement and its very solution. This is, obviously, one of the main issues of the software correctness problem.

The so-called formal methods for software development aim at clearly separating these two activities. A specification phase must provide a precise description of the problem to be solved and an implementation phase must yield a coded program fulfilling the given specification. In addition, it is assumed that this task may be carried out semi-automatically with the help of transformation tools.

Most work on formal methods concentrates on the implementation phase studying the transformation process and characterizing its correctness. However, specification building is not an easy activity. The specifier must depart, usually, from an incomplete and often inconsistent set of informal requirements and arrive to a formal specification fulfilling the badly expressed user needs. Therefore, the specifier, during this process, must clarify all the aspects of the problem to be solved by interacting with the user.

There is not much work done on specification development. The most common approach is the usual top-down refinement method: the problem to be specified is decomposed into subproblems, then, when all these subproblems have been completely specified an adequate combination of all the subspecifications, possibly together with some additional definitions, provides the specification for the whole problem. All specification languages provide support to this approach to specification design through the usual operations of enrichment and combination. The possibility of defining parameterized specifications, as in programming languages the possibility of defining procedures, may be seen as an abstraction facility to ease the refinement process.

This approach has a main drawback: at every moment it is assumed that the specifications we are dealing with are complete descriptions of the (sub)problems we are specifying, or, at least, complete descriptions with respect to some "unknown" parameter part. This is not realistic since every backtracking in the specification process, for instance due to user interaction or due to some inconsistency detection, may force to redo large parts of the specification under design. Modularity of the specification would not suffice to solve this problem for the very modular decomposition may be the result of some decisions taken to complete a given specification.

As for program design [GB80], specification design must be seen as a bidimensional refinement process. Horizontal refinements express the construction of a (not necessarily complete) specification of a given problem over the (also, possibly incomplete) specifications of the subproblems defined within the refinement process. That is, horizontal refinements can be formally seen as enrichments and extensions. Vertical refinements, as in program design, are the result of adding detail to, in this case completing, a given specification. Also as in program design it must be possible to compose or commute correct refinements to obtain, again, correct refinements [GB80].

In this paper we will present a category of specifications, together with their associated class of models, that has served as a basis for the definition of a specification language [CO88] built over these ideas. The key points of our approach are the following:

- Incompleteness is handled at the semantic level by means of looseness, i.e. an incomplete specification is a specification that has non-isomorphic models. Conversely, when a specification is complete then all its models are isomorphic.

- An incomplete specification may contain "parts" which are completely defined. This is handled by means of the concept of data, initial or free constraint [BG80, EWT83, Rei80]. That is, in our framework a specification SP is seen as a presentation P together with a set of constraints ζ which determine the unique (up to isomorphy) interpretation of the completely defined parts. The set of models of a specification is, as it can be expected, all P-algebras satisfying all constraints in ζ. In particular, when the constraints "define completely" all the sorts and operations of the given presentation then the semantics of SP is the initial P-algebra (Theorem 1.7). As it can be seen, though our approach is based on the loose semantics ideas [SW83], it is also strongly inspired by the initial semantics philosophy. We believe that an approach based on behavioural semantics would have been more adequate from a methodological standpoint. In particular, a notion of behaviour constraint based on the constructions introduced in [NO88] could have been used as a basis for such an approach. However, we have preferred to stick to the initial philosophy until some technical problems concerning the existence of amalgamations and extensions for certain classes of behaviour specification morphisms are solved [ONE88].

- Horizontal refinements are defined by means of *loose extensions*, that is extensions that preserve the class of models of the extended specification. On the other hand, vertical refinements are defined by means of refinement morphisms, a restriction of the specification morphisms used when dealing with constraints [Ehg88]. The usual results about composition of horizontal and vertical refinements are shown. Moreover, it is shown the compatibility of the specification level and the model level semantics of the specification building operations based on this kind of refinements.

A key notion of our approach, presented in this paper, is the concept of relative persistency of a constraint with respect to a given specification. Relative persistency is a generalization of the usual persistency notion. In this paper, we obtain results that generalize, with respect to this new notion, classical results concerning parameterized specifications and parameter passing [EM85]. In fact, in our approach there is no need for the concept of parameterized specification, since every incomplete specification may be seen as a specification implicitly parameterized by its non completely defined parts. Then, parameter passing would just consist in, applying the horizontal composition theorem (cf. theorem 2.7), defining explicitely the application of an induced vertical refinement on a loose extension. In particular, a parameterized specification in the usual sense [EM85] may be seen just as a special case of an incomplete specification with a constraint.

The main difference with other work on algebraic specifications with loose semantics [SW83, ST87a,ST87b] is our concern with respect to the proof-theoretical level of semantics. A consequence of this concern are the compatibility results shown in the paper. Also, we have tried to obtain proof-theoretical conditions for checking relative persistency, since this property is the basis for guaranteeing internal correctness of our specifications. It was clear from the very beginning that working with constraints would make almost impossible [BBTW81, MS85] getting a proof theoretical characterization of relative persistency, but we wanted to get reasonable sufficient conditions for guaranteeing it. We must admit that, concerning this aspect, the final results obtained were not as good as initially expected. We hope that the counter-examples that disclaimed our original hypothesis will give more insight on the problems of working with specifications with constraints. Anyhow, this does not mean that the conditions cannot be improved. On the contrary, at present we are working to provide some better results.

The paper is organized as follows: in the first section we present the basic concepts, including some results that will be used throughout the paper; the second section studies the notions of horizontal and vertical refinements together with their associated specification building operations; section three presents some proof-theoretical conditions for checking relative persistency; finally, in section four we establish some conclusions and we analyze related work.

ACKNOWLEDGEMENTS

The authors would like to thank H. Ehrig for the discussions we had at different stages of the writing of this paper that strongly influenced the final results.

1. Basic concepts

As we have said above, we foresee the process of specification development as a process of bidimensional refinement over incomplete specifications. To us a specification is incomplete if some parts of it have not been "completely defined", although some other parts may be. For instance, given the following specifications:

```
Val_eq = enrich Bool with sorts val
                          opns  eq: val x val → bool
                          eqns  eq(X,X) = true
                                eq(X,Y) = eq(Y,X)
                                (eq(X,Y) and eq(Y,Z)) ⊃ eq(X,Z) = true

Set = enrich val_eq defining sorts set
                             opns  ∅: set
                                   add: set x val → set
                                   _∈_ : val x set → bool
                             eqns  add(add(S,X),Y) = add(add(S,Y),X)
                                   add(add(S,X),X) = add(S,X)
                                   X∈ ∅ = false
                                   X∈ add(S,Y) = (X∈S) or eq(X,Y)

Choose = enrich Set with opns choose: set → val
                         eqns choose(S)∈S
```

the specification Choose may be considered incomplete since it is not "defined" what the values "are" or how the operation choose really works. However, the specification contains some parts which are "completely defined", namely the booleans and the sets, in this case depending on what the values are.

The main idea for dealing with this specification is to consider that its "complete parts" are constraints that impose how these parts should be in every admissible model. Then, to us, a specification will be a presentation together with a set of constraints.

1.1 Definitions

A presentation P is a tuple <Σ, E>, where Σ is a signature i.e. a pair <S, Op> where S is a se of sorts, Op a family of S-sorted operators and E a set of Σ-equations.

A constraint C is a pair of presentations (P1, P1') such that $P1 \subseteq P1'$. Given a presentation P, a constraint C = (P1, P1') is defined on P if $P1' \subseteq P$. An algebra $A \in Alg(P)$ satisfies a constraint (P1, P1') on P, denoted $A \models (P1, P1')$ iff

$$(A|_{P1})|^{P1'} \cong A|_{P1'}$$

Notation : $\_|_P : Alg(P') \to Alg(P)$ and $\_|^{P'} : Alg(P) \to Alg(P')$ denote, respectively, the forgetful functor and the free functor associated to the inclusion $P \subseteq P'$.

This means that a P-algebra satisfies the constraint (P1, P1') if the P1' part of the algebra is freely generated from the P1 part. For this reason, this is called a free generating, initial or data constraint [BG80, EWT83, Rei80]. In fact, our notion of constraint is a slight restriction of the notions defined in these papers, since they allow an arbitrary presentation morphism between P1' and P while we just allow inclusion.

1.2 Definition

A specification SP is a pair <P, ζ> where P is a presentation and ζ is a set of constraints on P. The semantics of a specification SP is defined by the following class of models

$$\mathrm{Mod}(SP) = \{A \in \mathrm{Alg}(P) \,/\, A \models \zeta\}$$

In the example above the specification we consider that the specification Choose consists of a presentation P containing all sorts, operations and equations shown in the example and two constraints (Ø,bool) and (val_eq, set) that impose that the models of this specification are algebras whose Boolean part is the boolean algebra of two elements and whose Set part are the finite sets of the elements of sort val, whatever they are.

A specification may be considered correct when it describes adequately the user needs. Obviously, it is impossible to formalize this kind of correctness notion. However, there are certain properties, like consistency, that we may want to be fulfilled by any specification. Often, it is said that a specification is (internally) correct if it satisfies some of these properties. In our case, we have taken the simplest correctness criteria, namely consistency of the given specification. In a previous paper [CO88] correctness was defined by a much more complicated notion. We tried to put in the definition not only the fact that a specification had models but also the fact that the specification had been "properly" built. We now think that this was, both, a methodological and a technical mistake since, on one hand, a specification can be correct even if its building process has not been what we would consider adequate, and, on the other, the complication of the concept also complicated all the technical apparatus.

1.3 Definition

A specification SP = <P, ζ> is correct iff it is consistent i.e. Mod(<P, ζ>) is not empty.

The following notion, relative persistency of a constraint with respect to a specification, is one of the key concepts of our approach. In [CO88] we asked all constraints to be persistent. This was far too restrictive since constraints (P1,P2) such that S2-S1 = Ø can hardly be persistent. For instance, the specification of the addition over natural numbers, defined by the constraint (Nat, Add), where Nat is the usual specification of the naturals and Add is the usual enrichment defining the addition, would not be persistent. On the other hand, it seems reasonable to be able to ask for some form of conservativeness when defining an extension over a given specification. With our notion the problem is solved since (Nat,Add) is persistent relative to any specification containing the constraint (∅,Nat). In fact, relative persistency is a generalization of the usual notions of conservative extension to be found in the literature.

1.4 Definition

Given a specification SP = <P, ζ>, a constraint C = (P1, P1'), with P1 ⊆ P, is <u>persistent relative to</u> SP iff for any A in Mod(SP):

$$(A|_{P1}|^{P1'})|_{P1} = A|_{P1}$$

A very important technical result that will be used throughout this paper is the following version of the Extension Lemma, which, as usual, provides some kind of compatibility between some form of extensions at the specification level and at the model level. In our case, the Extension Lemma states that if we extend a specification SP by means of a new constraint persistent relative to SP, then the semantics of the new specification is obtained as a construction of the semantics of SP and the semantics of the constraint. This will be stated more precisely in theorem 2.3.

1.5 Extension Lemma

Given a specification SP = <P, ζ> and a constraint C = (P1, P2) such that P1 ⊆ P, let P+P2 denote the result specification of the pushout diagram:

$$\begin{tikzcd}
P1 \arrow[r] \arrow[d] & P2 \arrow[d] \\
P \arrow[r] & P+P2
\end{tikzcd}$$

then if (P1,P2) is persistent relative to SP we have that (P, P+P2) is persistent relative to SP and the associated free functor $\_\,|^{P+P2}$: Alg(P) → Alg(P+P2) is an extension of $\_\,|^{P2}$: Alg(P1) → Alg(P2) for SP-models, that is for every A in Mod(SP):

$$A|_{P1}|^{P2} = A|^{P+P2}|_{P2}$$

Moreover, Mod(SP)$|^{P+P2}$ ⊆ Mod(<P+P2, ζ∪{(P1,P2)}>)

Proof

To prove the Lemma it is enough to show that for every A in Mod(SP):

$$A|^{P+P2} = A \oplus_{A|_{P1}} A|_{P1}|^{P2}$$

since, if this is true, clearly, the free functor $\_\,|^{P+P2}$: Alg(P) $\rightarrow$ Alg(P+P2) would be an extension of $\_\,|^{P2}$: Alg(P1) $\rightarrow$ Alg(P2) for SP-models and (P, P+P2) would be persistent relative to SP. In addition, by the amalgamation lemma for specifications with constraints [Ehg88], we would have (Mod(SP) $|^{P+P2}$) $\subseteq$ Mod(<P+P2, ζ+{(P1,P2)}>)

Therefore, we have to show that for each A in Mod(SP), each P+P2-algebra A' and each homomorphism h: A$\rightarrow$A' $|_P$, there is a unique homomorphism h' from $A'' = A \oplus_{A|_{P1}} A|_{P1}|^{P2}$ into A' such that the following diagram commutes:

$$\begin{tikzcd}
A \arrow[r, "id"] \arrow[dr, "h"'] & A''|_P \arrow[d, "h'|_P"] \\
& A'|_P
\end{tikzcd}$$

where id is the identity. Let A0 be A $|_{P1}$, A0' be A' $|_{P2}$ and h0 be h $|_{P1}$. That is, h0: A0 $\rightarrow$ A0' $|_{P1}$. Now, since (P1,P2) is persistent relative to SP, there is a unique homomorphism h0': A0 $|^{P2}$ $\rightarrow$ A0', such that the following diagram commutes:

$$\begin{tikzcd}
A0 \arrow[r, "id"] \arrow[dr, "h0"'] & A0|^{P2}|_{P1} \arrow[d, "h0'|_{P1}"] \\
& A0'|_{P1}
\end{tikzcd}$$

Then, we have that h $|_{P1}$ = h0 = h0' $|_{P1}$, therefore we may define h':

$$h' = h \oplus_{h0} h0'$$

by definition, we have that h' $|_P$ = h, hence we only have to show that h' is the unique homomorphism such that h' $|_P$ = h. Suppose h'' also satisfies that h'' $|_P$ = h, this would mean that h'' $|_{P2}$ is such that the following diagram commutes:

$$\begin{tikzcd}
A \arrow[r, "id"] \arrow[dr, "h"'] & A''|_P \arrow[d, "h''|_P"] \\
 & A'|_P
\end{tikzcd}$$

but, since $h''|_{P2}|_{P1} = h''|_P|_{P1} = h|_{P1}$, by freeness of $\_|^{P2}$, this means that

$$h''|_{P2} = h0' = h'|_{P2}$$

which, by unicity of the amalgamated sum, implies:

$$h' = h \oplus_{h0} h0' = h''$$

◆

Up to now, we have presented the basic semantic definitions of an approach for dealing with incomplete specifications. It seems reasonable to ask for compatibility of this semantics with the standard one for the case SP is "complete". In particular, in our case, this means that if SP is "complete" then Mod(SP) should coincide with the usual initial algebra semantics. This is indeed true, as we will see in theorem 1.7, but first let us define what does it mean for a specification to be complete.

1.6 Definition

A specification SP = <P, ζ> is <u>complete</u> iff the following two conditions hold:

a. <u>Complete definition:</u> $\forall s \in S\ \exists!(P1,P2) \in \zeta$ such that $s \in S2\text{-}S1$ and $\forall op \in Op\ \exists!(P1,P2) \in \zeta$ such that $op \in Op2\text{-}Op1$.

b. <u>No circularity:</u> The transitive closure of the relation <:
$(P1,P2) < (P3,P4) \Leftrightarrow \exists s \in S3$ such that $s \in S2\text{-}S1$ or $\exists op \in Op3$ such that $op \in Op2\text{-}Op1$
is a strict partial order on ζ.

Condition a. states that every sort and operation must be "completely defined" by a unique constraint. The uniqueness restriction is not really important but, without it, condition b. would have to be much more complicated and so would have to be the proof of the following theorem. Condition b. states that there is no circularity among the constraints. Absence of circularity is needed to consider a specification "complete". An example can show what can happen in the presence of circularity. Let P0, P0', P1, P1' and P be:

P0 = **Sorts** s1	P1 = **Sorts** s2	P = **Sorts** s1,s2
P0' = **Sorts** s1,s2	P1' = **Sorts** s1,s2	**Ops** f: s1 $\rightarrow$ s2
Ops f: s1 $\rightarrow$ s2	**Ops** g: s2 $\rightarrow$ s1	g: s2 $\rightarrow$ s1

and let SP be (P, {(P0,P0'),(P1,P1')}. It should be obvious that, although SP satisfies condition a., it should not be considered complete because there is no precise description of what sorts s1 and s2 should be like (apart of being a copy one of the other).

1.7 Theorem

Let $SP = <P, \zeta>$ be a consistent specification, then if SP is complete we have:

$$Mod(SP) = \{A \in Alg(P) / A \cong T_P\}$$

Proof

Let $C_0, C_1, ..., C_n$ be a topological sort of ζ with respect to the partial order defined by condition b. that is $C_i < C_j$ implies $i<j$. Note that C_0 must have the form $(\emptyset, P_0)$. Let $SP_0 = <P_0, \zeta_0>$, ..., $SP_n = <P_n, \zeta_n>$ be the following sequence of specifications:

$$SP_0 = <P_0, (\emptyset, P_0)>$$
$$SP_{i+1} = <P_{i+1}, \zeta_i \cup \{ (P'_{i+1}, P''_{i+1}) \}>$$

where $C_{i+1} = (P'_{i+1}, P''_{i+1})$ and P_{i+1} denotes the result of the pushout:

$$\begin{tikzcd}
P'_{i+1} \arrow[r] \arrow[d] & P''_{i+1} \arrow[d] \\
P_i \arrow[r] & P_{i+1}
\end{tikzcd}$$

We will prove by induction that for every i:

1. C_i is persistent relative to SP_{i-1}. In the case $i = 0$ we consider C_i to be persistent (persistent relative to the empty specification), which trivially is since $C_0 = (\emptyset, P_0)$.

2. $T_{P_i} \in Mod(SP_i)$

3. If A, $B \in Mod(SP_i)$ then $A \cong B$.

It should be clear that, if 1., 2. and 3. hold for every i, then the theorem is true since, by construction, $SP_n \subseteq SP$ and in addition, by condition a., $\Sigma_n = \Sigma$ and $\zeta_n = \zeta$. Then, $Mod(SP) \subseteq Mod(SP_n)$. But, if $Mod(SP_n)$ only contains algebras which are isomorphic to T_{Pn} and $Mod(SP)$ cannot be empty, since it is assumed to be consistent, then $Mod(SP) = Mod(SP_n)$ and $T_{Pn} \cong T_P$.

If $i = 0$ then, as it was said above, condition 1. trivially holds. Also, conditions 2. and 3. are obviously satisfied since the only P_0-algebras that satisfy the constraint $(\varnothing, P_0)$ are exactly the algebras which are isomorphic to T_{P0}.

Assume $i = j+1$. To prove that C_{j+1} is persistent relative to SP_j we have to prove that:

$$T_{Pj}|_{P'j+1}|^{P''j+1}|_{P'j+1} \cong T_{Pj}|_{P'j+1}$$

Now, if SP is consistent there should be an A such that $A \in Mod(SP)$, but since T_{Pj} is the only P_j-algebra satisfying the constraints in ζ_j, this means that $A|_{Pj} \cong T_{Pj}$. On the other hand, A must also satisfy the constraint C_{j+1}, therefore:

$$A|_{P'j+1}|^{P''j+1} \cong A|_{P''j+1}$$

but this implies that:

$$T_{Pj}|_{P'j+1}|^{P''j+1} \cong A|_{Pj}|_{P'j+1}|^{P''j+1} \cong A|_{P''j+1}$$

and therefore:

$$T_{Pj}|_{P'j+1}|^{P''j+1}|_{P'j+1} \cong A|_{P''j+1}|_{P'j+1} = A|_{P'j+1} \cong T_{Pj}|_{P'j+1}$$

Now, to prove 2. it is enough to notice that, since (P'_{j+1}, P''_{j+1}) is persistent relative to SP_j, according to the Extension Lemma $T_{Pj}|^{Pj+1}$ is in $Mod(SP_{j+1})$. But, $T_{Pj}|^{Pj+1} \cong T_{Pj+1}$.

Finally, 3 is also a consequence of the Extension Lemma. On one hand we have that all algebras in $Mod(SP_j)$ are isomorphic which implies that all algebras in $Mod(SP_j)|_{P'j+1}$ are also isomorphic and, therefore, so it happens with algebras in $Mod(SP_j)|_{P'j+1}|^{P''j+1}$. On the other, from the Extension Lemma we have that:

$$Mod(<P_{j+1}, \zeta_j \cup \{C_{j+1}\}>) = Mod(SP_j) \oplus_{Alg(P'j+1)} (Alg(P'_{j+1})|^{P''j+1})$$

then, from the Amalgamation Lemma [EM85], we have that all algebras in $Mod(<P_{j+1}, \zeta_j \cup \{C_{j+1}\}>)$ are also isomorphic. ◆

2. Specification building operations

In this section, we will present our notions of horizontal and vertical refinement for specification development. Together with this notions we will introduce the basic operations for defining this refinements.

Horizontal refinements are defined by the notion of loose extension, which is a generalization of the notion of conservative extension.

2.1 Definition

Given specifications SP1 and SP2, SP2 is a loose extension of SP1 iff

a. $SP1 \subseteq SP2$
b. $\mathrm{Mod}(SP1) = \mathrm{Mod}(SP2)|_{P1}$

We consider three basic operations for defining loose extensions: **enrich defining, enrich with** and **combine.**

The operation **enrich defining** adds to a given specification new sorts and operations together with a constraint defining them. That is, given a specification SP = $<P, \zeta>$ and a constraint C = (P1,P2) such that $P1 \subseteq P$, **enrich defining** creates a new specification $<P+P2, \zeta \cup \{C\}>$, where P+P2 denotes, as in the Extension Lemma (cf. 1.5) the pushout of P and P2 over P1.

The operation **enrich with** adds new sorts and operations without any new constraint. That is, given a specification SP = $<P, \zeta>$, where P = ((S,Op),E), and a triple (S1,Op1,E1), such that P1 = ((S+S1,Op+Op1), E+E1) is a presentation and where + denotes disjoint union, **enrich with** creates the new specification $<P1,\zeta>$.

Finally, the operation **combine** puts together two specifications whithout duplicating their common part. That is, given specifications SP1, SP2 and SP3, such that SP2 and SP3 are loose extensions of SP1, the combination of SP2 and SP3 over SP1 is defined as the result of the pushout:

$$\begin{tikzcd}
SP1 \arrow[r] \arrow[d] & SP2 \arrow[d] \\
SP3 \arrow[r] & SP4
\end{tikzcd}$$

The semantics of these three operations could also be defined at the model level by means of amalgamated sums. This model level definitions would be compatible with the previous ones, since, by the Amalgamation Lemma for specifications with constraints [Ehg88], under the conditions stated above for the three operations, we know that:

$\mathrm{Mod}(<P+P2, \zeta\cup\{(P1,P2)\}>) = \mathrm{Mod}(<P, \zeta>) \oplus_{\mathrm{Mod}(<P1, \varnothing>)} \mathrm{Mod}(<P2, \{(P1,P2)\}>)$
$\mathrm{Mod}(<P1, \zeta>) = \mathrm{Mod}(<P, \zeta>) \oplus_{\mathrm{Mod}(<P, \varnothing>)} \mathrm{Mod}(<P1, \varnothing>)$
$\mathrm{Mod}(SP4) = \mathrm{Mod}(SP2) \oplus_{\mathrm{Mod}(SP1)} \mathrm{Mod}(SP3)$

In what follows, we will study the correctness of these three operations, i.e. under which conditions these operations define loose extensions. The simplest case is the **combine** operation, since the result SP4 of the combination of two specifications, SP2 and SP3, that are loose extensions of SP1 is always a loose extension of SP2 and SP3:

2.2 Theorem

Let SP1, SP2 and SP3 be three consistent specifications such that SP2 and SP3 are loose extensions of SP1 and let SP4 be the result of the pushout:

$$\begin{tikzcd}
SP1 \arrow[r] \arrow[d] & SP2 \arrow[d] \\
SP3 \arrow[r] & SP4
\end{tikzcd}$$

then SP4 is a loose extension of SP2 and SP3.

Proof

The proof is almost trivial: wolog, let us prove that SP4 is a loose extension of SP2. Let A2 be in Mod(SP2), then we know that $A1 = A2|_{P1}$ is in Mod(SP1) and, since SP3 is an extension of SP1, there should be an A3 in Mod(SP3) such that $A1 = A3|_{P1}$. Then, by the Amalgamation Lemma for specifications with constraints [Ehg88], we have that $A4 = A2 \oplus_{A1} A3$ is in Mod(SP4).

The case of **enrich defining** is also quite simple. It depends on the relative persistency of the new constraint with respect to the enriched specification:

2.3 Theorem

Given a specification SP = <P, ζ> and a constraint C = (P1, P2) such that $P1 \subsetneq P$ and let P+P2 be the result of the pushout:

$$\begin{tikzcd}
P1 \arrow[r] \arrow[d] & P2 \arrow[d] \\
P \arrow[r] & P+P2
\end{tikzcd}$$

then SP' = <P+P2, $\zeta \cup \{C\}$> is a loose extension of SP iff C is persistent relative to SP.

Proof

If SP' is a loose extension of SP this means that for every P-algebra A such that $A \models \zeta$ there is a P+P2-algebra B such that $B \models \zeta \cup \{C\}$ and $B|_P = A$. Now, if B satisfies C this means that:

$$B|_{P1}|^{P2} = B|_{P2}$$

but this implies that:

$$B|_{P1}|^{P2}|_{P1} = B|_{P2}|_{P1} = B|_{P1}$$

and therefore:

$$A|_{P1}|^{P2}|_{P1} = B|_{P1}|^{P2}|_{P1} = B|_{P1} = A|_{P1}$$

Conversely, if (P1,P2) is persistent relative to SP, by the Extension Lemma proved above, we know that for every $A \in \mathrm{Mod(SP)}$ it holds that $A|^{P+P2} \in \mathrm{Mod(SP')}$ and $A|^{P+P2}|_P = A$. ◆

Finally, the correctness of the **enrich with** operation is the most complicated case. Here we will just give a sufficient condition which we think can handle many situations. Essentially, it says that an enrichment of this kind over a specification SP is a loose extension if we can provide a constraint persistent relatively to SP, "defining completely" the enrichment. We think that this is a reasonable condition for many situations since, often, the reason of adding some sorts or operations without defining them completely is that we do not want to take a decision of choosing among several possible alternatives.

2.4 Corollary

Given a specification SP=<P,ζ>, and a presentation P1, such that $P \subseteq P1$, then SP1=<P1,ζ> is a loose extension of SP if there exists a constraint C=(P2,P3), such that $P2 \subseteq P$, $P1 \subseteq P3$ and (P2,P3) is persistent relative to SP.

Proof

If there is a constraint (P2,P3) such that $P2 \subseteq P$, $P1 \subseteq P3$ and (P2,P3) is persistent relative to SP, then using the previous theorem we know that SP' = <P+P3, $\zeta \cup \{C\}$> is a loose extension of SP. But, $\mathrm{Mod(SP')}|_{P1} \subseteq \mathrm{Mod(SP1)}$ thus SP1 is a loose extension of SP. ◆

The second kind of refinements we consider are vertical refinements. A vertical refinement consists on "adding detail" to a specification, in our case completing the given specification or, similarly, restricting its class of models. In this sense, it seems reasonable to consider vertical refinements as some class of specification morphism. We have considered a definition which is more restrictive than it, perhaps, could be. In particular, we have restricted *refinement morphisms* to translate constraints injectively. The reason for this is, mainly, methodological. According to our approach a constraint represents a part of a specification completely defined. In this sense, it seems reasonable to think that when we are completing a specification the already completed parts should remain "untouched". A similar restriction is taken in [ETLZ82] but, apparently, just for technical reasons.

2.5 Definition

A refinement morphism f: <P1,ζ1> → <P2,ζ2> is a presentation mophism f: P1 → P2, satisfying:

a. f is injective on constrained sorts and operations, that is for every constraint (P,P') in ζ1, if s1,s2∈S'-S (resp. op1,op2∈Op'-Op) then f(s1) = f(s2) implies s1 = s2 (resp. f(op1) = f(op2) implies op1 = op2).

b. f(ζ1) ⊆ ζ2.

2.6 Facts

1. Loose extensions may be seen as special cases of refinement morphisms.

2. Obviously, the composition of vertical refinements is a vertical refinement. Therefore, vertical composition trivially holds.

3. If f: <P1,ζ1> → <P2,ζ2> is a refinement morphism then $\mathrm{Mod}(\langle P2,\zeta 2\rangle)|_{P1} \subseteq \mathrm{Mod}(\langle P1,\zeta 1\rangle)$. This is a consequence of the restriction imposing f to be injective on ζ1.

4. There are pushouts (amalgamations) associated to categories of specifications (models) with refinement morphisms (the associated forgetful functors). That is, given specifications SP1 = <P1,ζ1>, SP2 = <P2,ζ2> and SP3 = <P3,ζ3> and refinement morphisms f0: <P1,ζ1> → <P2,ζ2> and f1: <P1,ζ1> → <P3,ζ3> we may define the pushout:

$$\begin{tikzcd}
SP1 \arrow[r, "f0"] \arrow[d, "f1"'] & SP2 \arrow[d, "f2"] \\
SP3 \arrow[r, "f3"'] & SP4
\end{tikzcd}$$

and in addition, $\mathrm{Mod}(SP4) = \mathrm{Mod}(SP3) \oplus_{\mathrm{Mod}(SP1)} \mathrm{Mod}(SP2)$.

The main operation for defining vertical refinements is presented in the following theorem. In particular, it shows how we can substitute, within a specification, an incomplete part for a more complete one. Specifically, it states how a vertical refinement of a given specification SP1 induces a vertical refinement on any loose extension of SP1. This fact has several interpretations. On one hand, the theorem states, in our framework, the horizontal composition property [GB80], namely, that the order in which we perform vertical and horizontal refinements is not important. On the other, it shows that in our framework there is no need for parameterization, since any specification SP2 may be seen as having as implicit parameters all specifications SP1 loosely extended by SP2. Then, this induced vertical refinement may be seen as a generalized form of parameter passing. The relation of our construction to parameter passing is very similar to the one found by B. Meyer [Mey86], at the programming language level, between genericity and inheritance, showing that inheritance may be seen as a generalization of genericity. Indeed, as it is shown in [CO88], our notion of vertical refinement may be seen, from a methodological standpoint, as an inheritance relation defined at the specification level. Obviously, this kind of inheritance relation has nothing to do with the subtyping (or subsorting) relation also studied in the literature [GM83].

2.7 Theorem

Let SP1, SP2 and SP3 be consistent specifications such that SP2 is a loose extension of SP1 and f: SP1 → SP3 is a refinement morphism. The result of substituting SP1 by SP3 in SP2 is the specification SP4 = <P4,ζ4> defined by the pushout:

$$\begin{tikzcd}
SP1 \arrow[r] \arrow[d, "f"'] & SP2 \arrow[d, "f'"] \\
SP3 \arrow[r] & SP4
\end{tikzcd}$$

then we have:

1. SP4 is a consistent specification.
2. SP4 is a loose extension of SP3

Proof

1. is an immediate consequence of 2., since if SP4 is a loose extension of SP3 and Mod(SP3) is not empty then Mod(SP4) is not empty.

2. We know that

$$\mathrm{Mod}(SP4) = \mathrm{Mod}(SP3) \oplus_{\mathrm{Mod}(SP1)} \mathrm{Mod}(SP2)$$

Now, given an $A \in \mathrm{Mod}(SP3)$, we have that $A|_{P1} \in \mathrm{Mod}(SP1)$. But, if SP2 is a loose extension of SP1, there is a $B \in \mathrm{Mod}(SP2)$ such that $B|_{P1} = A|_{P1}$. Then, defining:

$$B' = A \oplus_{A|_{P1}} B$$

we have that $B' \in \mathrm{Mod}(SP4)$ and $B'|_{P3} = A$. ◆

In the previous theorem, the fact that SP2 is a loose extension of SP1, i.e. $\mathrm{Mod}(SP2)|_{P1} =$

Mod(SP1), is absolutely needed to guarantee the consistency of SP4. The situation is similar to the need of persistency to assure the correctness of parameter passing:

2.8 Theorem

Let SP1 and SP2 be specifications such that SP1 $\subseteq$ SP2, then if SP2 is not a loose extension of SP1 there is a specification SP3 and a refinement morphism f: SP1 $\rightarrow$ SP3 such that the result, SP4, of the associated pushout diagram:

$$\begin{tikzcd} SP1 \arrow[r] \arrow[d, "f"'] & SP2 \arrow[d, "f'"] \\ SP3 \arrow[r] & SP4 \end{tikzcd}$$

is not consistent.

Proof

If SP2 is not a loose extension of SP1 this means that there is an A1$\in$Mod(SP1) such that for every A2$\in$Mod(SP2) A2$|_{P1}$ is not isomorphic to A1. Let SP3 = <P3, ζ3>, where P3 is the presentation obtained by adding to P1 all the values from A1 as constants of the appropriate sorts and all the equations satisfied by A1, and ζ3 is obtained by adding to ζ1 the constraint ($\varnothing$,P3). Obviously, Mod(SP3) = {B$\in$Alg(P3) / B$|_{P1} \cong$ A1}.

Suppose SP4 is consistent, this would mean that there is an A4 in Mod(SP4). But then A4$|_{P3}$$\in$Mod(SP3), i.e. A4$|_{P3}|_{P1} \equiv$ A1. However, this would mean that A4$|_{P2}|_{P1} \equiv$ A1, hence A4$|_{P2}$ would be a model in Mod(SP2) extending A1. ◆

3. Relative persistency: proof-theoretical results

As we have seen, most of our correctness results depend on the property of relative persistency. Therefore, if we are interested in the practical use of our ideas we should provide ways of checking this property. The main problem is that a complete proof system cannot exist for specifications with constraints [BBTW81,MS85]. Hence we cannot expect to obtain conditions, based on equational deduction, equivalent to relative persistency. However, we thought that it was possible to obtain such a characterization in terms of the theories defined by the specifications. The idea was that, once a necessary and sufficient condition (in terms of these theories) was obtained, a relaxation of the condition substituting the theories by the equational theories associated to the presentations would provide reasonable sufficient conditions that could be checked using the usual deductive tools associated to equational specifications. In this sense, we defined hierarchical consistency and sufficient completeness conditions that seemed adequate to characterize (as for the usual persistency) relative persistency. Our conjecture proved to be wrong as we will see, but first let us introduce some notation.

Notation

Given a specification $SP = <P, \zeta>$ and an algebra $A \in Mod(SP)$ we define:
$CS(<P, \zeta>) = \{s \in S / \exists(P1,P2) \in \zeta$ such that $s \in S2-S1\}$
$NCS(<P, \zeta>) = S - CS(<P, \zeta>)$
$C\Sigma(<P, \zeta>) = \{op \in \Sigma / \exists(P1,P2) \in \zeta$ such that $op \in \Sigma2-\Sigma1\}$
$NC\Sigma (<P, \zeta>) = \Sigma - C\Sigma(<P, \zeta>)$
$Th(SP) = \{t1 = t2/ t1,t2 \in T_\Sigma(X)$ such that $\forall A \in Mod(SP)\ A \models t1=t2\}$
$Eqns(A) = \{t1 = t2/ t1,t2 \in T_\Sigma(A)$ such that $A \models t1=t2\}$.

In what follows, we will assume that the specifications we are dealing with satisfy the no circularity condition of definition 1.6. This assumption is technically needed to obtain the results of this section. Anyhow we could lifted it just defining CS(SP) as the set of sorts that are defined by "not circular" constraints, i.e. constraints for which the order relation defined in 1.6 is not reflexive.

3.1 Definition

Given a specification $SP = <P, \zeta>$ and a constraint $C = (P1, P2)$ such that $P1 \subseteq P$ and $(P2-P1) \cap P = \varnothing$ then:

a. C is sufficiently complete relative to SP iff $\forall t1 \in T_{C\Sigma(SP)+\Sigma2}(X_{NCS(SP)})$ of sort in S1 $\exists t2 \in T_\Sigma(X_{NCS(SP)})$ such that $Th(P, \zeta)+E2 \vdash t1=t2$.

b. C is hierarchy consistent relative to SP iff $\forall t1,t2 \in T_{C\Sigma(SP)+\Sigma1}(X_{NCS(SP)})$ we have that $Th(P, \zeta)+E2 \vdash t1=t2$ iff $Th(P, \zeta) \vdash t1=t2$.

These conditions generalize the usual conditions that characterize persistency in the standard case [Gan83]. Relative sufficient completeness tries to characterize the "no junk" condition, i.e. that the free functor associated to (P1,P2) does not introduce "junk" on sorts from S1 in algebras from Mod(SP). The idea of this condition is that the possible junk may be represented by terms of sort in S1 that contain operations from Σ2, hence we should ask that these junk candidates should be equivalent to non-junk terms. However, in the standard case, these terms are over variables of sort in S1, because we assume that we can have any sets of values as carriers for these sorts. This is not true anymore, for us the "unknown" sorts are the unconstrained sorts, i.e. these belonging to NCS(SP). The values of constrained sorts, i.e. these belonging to CS(SP), are generated by terms in $T_{C\Sigma(SP)}(X_{NCS(SP)})$. For this reason, we consider as candidates for generating junk the terms in $T_{C\Sigma(SP)+\Sigma2}(X_{NCS(SP)})$.

Relative hierarchy consistency tries to characterize the usual "no confusion" condition, i.e. that the free functor associated to (P1,P2) does not identify values from sorts from S1 in algebras from Mod(SP). The idea is to state that the added equations cannot make deducibly equal two terms, representing values of the possible models, that were not already equivalent. For the same reasons as for sufficient completeness, the class of terms that we consider that represent those values is $T_{C\Sigma(SP)+\Sigma1}(X_{NCS(SP)})$.

Unfortunately, as we said above, these conditions do not really characterize relative persistency. In what follows we will see that relative sufficient completeness is just a sufficient condition for the no-junk property, and relative hierarchy consistency is just a necessary condition for the no-confusion property.

3.2 Lemma

Given a specification SP=<P,ζ> and an algebra A in Mod(SP) then $\forall a\in A_s$, with $s\in CS(SP)$, $\exists t\in T_{C\Sigma(SP)}(A_{NCS(SP)})$ such that A |= a = t.

Proof

We will prove the lemma by noetherian induction on the order defined among constraints in the non-circularity condition. Let $a\in A_s$, with $s\in CS(SP)$ and let C = (P1,P2) be the constraint such that $s\in S2\text{-}S1$, then, since A|= C, there should exist a term $t1\in T_{\Sigma 2}(A|_{P1})\cap T_{C\Sigma(SP)}(A|_{P1})$ such that A |= a = t1. Now, all constrained sorts in S1 are "defined" in constraints smaller than C, then using noetherian induction $\forall b\in A_s$, with $s\in CS(SP)\cap S1$, $\exists t\in T_{C\Sigma(SP)}(A_{NCS(SP)})$ such that A |= b = t. Let us select one of such terms for every b and call it t_b. Let σ be the substitution $\sigma: A_{CS(SP)\cap S1}\rightarrow T_{C\Sigma(SP)}(A_{NCS(SP)})$, defined for every b, $\sigma(b) = t_b$. Then, we have that A |= σ(t1) = a and $\sigma(t1)\in T_{C\Sigma(SP)}(A_{NCS(SP)})$. ◆

3.3 Proposition

If C = (P1,P2) is sufficiently complete relative to SP then for every $A0\in Mod(SP)$ it holds that, for $A = A0|_{P1}$, the unit $\varepsilon_A: A \rightarrow A|^{P2}|_{P1}$ is surjective.

Proof

We have to see that for every A in Mod(SP) and for every $t1\in T_{\Sigma 2}(A_{S1})$ with sort in S1 there is a $t2\in T_{\Sigma 1}(A_{S1})$ such that $A|^{P2}|_{P1}$|= t1 = t2 or, equivalently, Eqns(A)+E2 |- t1 = t2.

By the previous lemma we know that there is a $t3\in T_{C\Sigma(SP)+\Sigma 2}(A_{NCS(SP)})$ such that Eqns(A) |- t1 = t3. On the other hand, if C is sufficiently complete relative to SP then there exists a $t2\in T_{\Sigma}(A_{NCS(SP)})$ such that E2 |- t3 = t2. Therefore, Eqns(A)+E2 |- t1 = t2. ◆

The converse is not true:

3.4 Counter-example:

Let Nat be the usual presentation of the naturals (with 0 and suc) and let P0, P1 and P2 be the following presentations:

```
P0 = Nat + opns a: nat    P1 = P0 + opns f: nat → nat        P2 = P0 + opns g: nat → nat
                                    eqns                               eqns
                                    f(0) = 0                           g(0) = 0
                                    f(suc(0)) = 0                      g(suc(0)) = suc(0)
                                    f(suc(suc(x)) = suc(0)             g(suc(suc(x)) = g(a)
                                    f(a) = 0
```

Now, let SP be the specification <P1,{(∅,Nat),(P0,P1)}>, we have that the constraint (P0,P2) is not sufficiently complete relative to SP but it is persistent relative to SP. Being specific, (P0,P2) is not sufficiently complete relative to SP since from the equations and the theory of SP we cannot deduce that the term g(suc(suc(0)) is equivalent to any P1-term. On the other hand, (P0,P2) is persistent relative to SP since the SP-models are algebras whose values are the natural numbers, and the value of the constant a is either 0 or 1 (the possible interpretation of the other operations are not of interest here) and, on the other hand, the constraint (P0,P2) defines a new function g, on these models, such that either g(n) = **if** n = 1 **then** 1 **else** 0 or either g(n) = **if** n = 0 **then** 0 **else** 1, depending on the value of a in that model. ◆

3.5 Proposition

If for every $A0 \in Mod(SP)$ it holds that, for $A = A0|_{P1}$, the unit $\varepsilon_A: A \to A|^{P2}|_{P1}$ is injective then C is hierarchy consistent relative to SP.

Proof

Let t1 and t2 be two terms in $T_{C\Sigma(SP)+\Sigma1}(X_{NCS(SP)})$ such that $Th(SP) \nvdash t1=t2$. This means that there is an A0 in Mod(SP) and a substitution $\sigma: X_{NCS(SP)} \to A0$, such that $A0 \not\models \sigma(t1)=\sigma(t2)$ and, thus, $A0|_{P1} \not\models \sigma(t1)=\sigma(t2)$. Now, if $Th(SP)+E2 \vdash t1=t2$ this means that $Eqns(A0)+E2 \vdash \sigma(t1)=\sigma(t2)$, but then this implies that $A0|_{P1}|^{P2} \models \sigma(t1)=\sigma(t2)$. ◆

The converse is not true, even assuming that C is sufficiently complete relative to SP.

3.6 Counter-example:

Let P0, P1 and P2 be the following presentations:

```
P0 =  sorts s          P1 = P0 + opns a: s      P2 = P1 + opns f: s → s
      opns 0: s                                           eqns
           1: s                                           f(0) = 0
                                                          f(1) = 0
                                                          f(a) = 1
```

Now, let SP be the specification <P1,{(∅,P0)}>, we have that the constraint (P1,P2) is, trivially, hierarchy consistent and sufficiently complete relative to SP but it is not persistent relative to SP, in fact, the specification <P2,{(∅,P0)}> is inconsistent. ◆

Now, if we use a stronger notion of relative sufficient completeness we can prove that consistency is equivalent to no confusion. The problem is that this kind of sufficient completeness is too strong if the constraint C is such that S2 - S1 = ∅.

3.7 Definition

C = (P1,P2) is strongly sufficiently complete relative to SP iff $\forall t1 \in T_{\Sigma2}(X_{S1})$ of sort in S1 $\exists t2 \in T_{\Sigma1}(X_{S1})$ such that $Th(SP)+E2 \vdash t1=t2$.

3.8 Theorem

If C = (P1,P2) is strongly sufficiently complete relative to SP then C is persistent relative to SP iff C is hierarchy consistent relative to SP.

Proof

We have to prove that for any $A \in Mod(SP)$ and for any $t,t' \in T_{\Sigma 1}(A)$ if $A0 \mid_{P1} \mid^{P2} \models t = t'$ then $A \mid_{P1} \models t = t'$. This is equivalent to see that if there exist terms t0, ..., $tn \in T_{\Sigma 2}(A)$ such that t = t0, t'=tn and for any i $(0 \leq i < n)$ $ti \leftrightarrow_{E2+Eqns(A)} ti+1$ (i.e. we can obtain ti+1 from ti by rewriting using an equation from E2 or from Eqns(A)) then Eqns(A)|- t =t'. We will prove, by induction, that there are terms t0',..., $tn' \in T_{\Sigma}(A)$ such that t = t0', t'=tn' and for any i $(0 \leq i < n)$ it holds

a) Eqns(A) |- ti' =ti+1'

and b) E2 |- ti =ti'.

By Lemma 3.2, we know that for any $a \in A_s$, such that s is constrained, there is a $t \in T_{C\Sigma(SP)}(A_{NCS(SP)})$ such that Eqns(A) |- t = a. Let us select for any $a \in A_s$ one such t and let us denote it t_a and let σ be the substitution from $A_{CS(SP)}$ to $T_{C\Sigma(SP)}(A_{NCS(SP)})$ defined σ(a) = t_a.

Now, we proceed to the definition of the terms ti'. According to the above statement, t0' is already defined. To define ti+1' we consider two cases:

1) $ti \leftrightarrow_{E2} ti+1$. Using that C is strongly sufficiently complete relative to SP we define ti+1' as the term in $T_{\Sigma 1}(A)$ such that E2 |- ti+1 =ti+1' (in case ti+1 is already in $T_{\Sigma 1}(A)$ we define ti+1' as ti+1; this is enough to guarantee that tn'=t'). Hence, b) is already proved by construction. To prove a) it is enough to note that if E2 |- ti =ti', E2 |- ti =ti+1 and E2 |- ti+1 =ti+1', then E2 |- ti' =ti+1'. But this means that E2 |- σ(ti') = σ(ti+1'), with $\sigma(ti'),\sigma(ti+1') \in T_{C\Sigma(SP)}(A_{NCS(SP)})$. Then, using the fact that C is hierarchy consistent relative to SP, we have that E1+Th(SP) |- σ(ti') = σ(ti+1'), i.e. Eqns(A)|- σ(ti') = σ(ti+1'). But this implies Eqns(A) |- ti' =ti+1', for Eqns(A) |- ti' = σ(ti') and Eqns(A) |- ti+1' = σ(ti+1').

2) $ti \leftrightarrow_{Eqns(A)} ti+1$. This means that there is a position u in ti and an equation l=r in Eqns(A) such that $ti|_u$ = l and ti[u←r] = ti+1. Let x be a variable, let t0" be the term ti[u←x] and let σ1 and σ2 be two substitutions associating to x the terms l and r, respectively, i.e. σ1(t0") = l and σ2(t0") = r. Using the fact that C is strongly sufficiently complete relative to SP, there is a t1" in $T_{\Sigma 1}(A \cup \{x\})$ such that E2 |- t0"=t1" (in case t0" is already in $T_{\Sigma 1}(A \cup \{x\})$ we define t1" as t0"). Now, we define ti+1' as σ2(t1"). By construction we have that E2 |-ti+1=ti+1'. In addition, also by construction, if ti+1 is in $T_{\Sigma 1}(A)$ then ti+1'=ti+1 and, therefore, this guarantees tn'= t'. Thus, we have to prove that Eqns(A)|- ti'= ti+1'. Now, we know that E2|- ti'= σ1(t0"), since E2|- ti'= ti and E2|-ti=σ1(t0"), and this means that E2|- σ(ti') = σ(σ1(t0")), with $\sigma(ti'), \sigma(\sigma 1(t0")) \in T_{C\Sigma(SP)}(A_{NCS(SP)})$ and, therefore, using the fact that C is hierarchy consistent relative to SP, we have that E1+Th(SP) |- σ(ti') = σ(σ1(t0")), i.e. Eqns(A)|- σ(ti') = σ(σ1(t0")), but this implies Eqns(A) |- ti' = σ1(t0"), for Eqns(A) |- ti' = σ(ti') and Eqns(A) |- σ1(t0") = σ(σ1(t0")). Finally,

we have that Eqns(A) $|-$ $\sigma1(t0'') = \sigma2(t0'')$ and, thus, Eqns(A) $|-$ ti'=ti+1'. ◆

From the previous definitions and results we may provide "provable" conditions, namely conditions that, instead of being stated in terms of the theory associated to SP, are stated in terms of equational deduction from P.

3.9 Definition

Given a specification SP = <P, ζ> and a constraint C = (P1, P2) such that $P1 \subseteq P$ and $(P2-P1) \cap P = \emptyset$ then:

C is provably sufficiently complete relative to SP iff $\forall t1 \in T_{C\Sigma(SP)+\Sigma2}(X_{NCS(SP)})$ of sort in S1 $\exists t2 \in T_{\Sigma}(X_{NCS(SP)})$ such that E+E2 $|-$ t1=t2.

C is sufficiently complete iff $\forall t1 \in T_{\Sigma2}(X_{S1})$ of sort in S1 $\exists t2 \in T_{\Sigma1}(X_{S1})$ such that E2 $|-$ t1=t2.

C is provably hierarchy consistent relative to SP iff $\forall t1,t2 \in T_{C\Sigma(SP)+\Sigma1}(X_{NCS(SP)})$ we have that E+E2 $|-$ t1=t2 iff E $|-$ t1=t2.

The relation between provably sufficient completeness and no junk is just a direct consequence of 3.3, since if C is provably sufficiently complete relative to SP then C is sufficiently complete relative to SP. The case for consistency is not quite the same. It is not clear whether if C is provably hierarchy consistent relative to SP then C is hierarchy consistent relative to SP. However, proposition 3.11 can be proved without problems by means of a proof identical to the one of theorem 3.8.

3.10 Corollary

If C = (P1,P2) is provably sufficiently complete relative to SP then for every $A0 \in Mod(SP)$ it holds that, for $A = A0|_{P1}$, the unit $\varepsilon_A: A \rightarrow A|^{P2}|_{P1}$ is surjective.

3.11 Proposition

If C = (P1,P2) is sufficiently complete then C is persistent relative to SP iff C is hierarchy consistent relative to SP.

4. Conclusions and related work

We have presented a formal framework for the development of specifications. The main issues concerning the approach are the following:

- We allow to deal with incomplete specifications during the design process. Technically, this is handled by the use of initial constraints and loose semantics.

- As for programs, the design process is presented as a two-dimensional refinement process. Here, vertical refinements consist on completing the refined specification.

- The usual composition properties for refinements hold in our framework. In addition, the horizontal composition theorem defines a generalization of parameter passing as it is usually horizontal composition theorem defines a generalization of parameter passing as it is usually presented.

- Specification building operations have been defined according to both kinds of refinements. Moreover, there is compatibility between the specification level and the model level semantics of these operations.

- Proof-theoretic conditions for assuring that the specification building operations are correctness-preserving, in the sense that when applied under adequate conditions to correct (consistent) specifications they yield correct (consistent) specifications, have been established. These conditions are, in some cases, not sufficiently good. However, we expect that some better results could be obtained by imposing some restrictions on specifications.

Many of the methodological ideas of this paper have been heavily influenced by the work on loose semantics and program (specification) design done by Sannella, Wirsing and Tarlecki (e.g. [ST87a, ST87b, SW83]). However, the main difference between both approaches is our concern with proof-theory. In their framework, all the specification building operations are defined only at the model level, without any reference to the specification level. This simplification makes almost impossible checking the correctness of a given specification. On the contrary, we have all the time worked with full compatibility between the specification and the model levels so that internal or external correctness of a given specification can be tested. In particular, internal correctness can be checked by proving the conditions that assure the consistency of a specification. External correctness can be tested by prototyping, for instance by term rewriting.

To end, it must be said that, technically, this paper may be seen as a continuation of [ETLZ82]. In that paper the language Look was introduced based on specifications as ours. Also, parameter passing was generalized to a substitution operation as ours. However they were unable to obtain the kind of compatibility we do. In fact, this was posed as an open problem at the conclusions of that paper. Also, no attempt was made to provide persistency-like correctness conditions or to characterize them proof-theoretically.

5. References

[BBTW81] Bergstra, J.A.; Broy, M.; Tucker, J.V.; Wirsing, M. On the power of algebraic specifications, Proc. MFCS 1981, Springer LNCS 118, pp. 193-204.

[BG80] Burstall, R.M.; Goguen, J.A. The semantics of Clear, a specification language, Proc. Copenhagen Winter School on Abstract Software Specification, Springer LNCS 86, pp. 292-332, 1980.

[CO88] Clerici, S.; Orejas, F. GSBL: an algebraic specification language based on inheritance, Proc. Europ. Conf. on Object Oriented Programming, (Oslo, 1988), Springer LNCS.

[Ehg88] Ehrig, H. A categorical concept of constraints for algebraic specifications, T.U. Berlin November 1988.

[EWT82] Ehrig, H., Wagner, E.G. Thatcher, J.W. Algebraic constraints for specifications and canonical form results, Institut für Software und Theoretische Informatik, T.U. Berlin Bericht Nr. 82-09, 1982.

[EM85] Ehrig, H., Mahr, B.Fundamentals of algebraic specification 1, EATCS Monographs on Theor. Comp. Sc., Springer Verlag, 1985.

[ETLZ82] Ehrig, H., Thatcher, J.W., Lucas, P., Zilles, S.N. Denotational and initial algebra semantics of the algebraic specification language LOOK, Draft Report, IBM Research, 1982.

[Gan 83] Ganzinger, H. Parameterized specifications: parameter passing and implementation with respect to observability, TOPLAS 5,3 (1983), pp. 318-354.

[GB80] Goguen, J.A., Burstall, R.M. CAT, a system for the structured elaboration of correct programs from structured specifications, Tech. Report CSL-118, Comp. Sc. Lab., SRI Int., 1980.

[GM83] Goguen, J.A., Meseguer, J. Order-sorted algebra I: partial and overloaded operations, errors and inheritance. Technical Report, SRI International, Computer Science Lab 1983.

[MS85] MacQueen D.B., Sannella, D.T. Completeness of proof systems for equational specifications. IEEE Trans. on Software Engineering, SE-11(5), 454-461 (1985).

[Mey86] Meyer B. Genericity versus Inheritance, Proc. ACM conf. Object-Oriented Programming Syst, Languages, and Applications, ACM, New York, 1986, pp. 391-405

[NO88] Nivela, P.; Orejas, F. Initial behaviour semantics, in 'Recent Trends in Algebraic Specification', D. Sannella, A. Tarlecki (eds.), Springer LNCS , 1988.

[ONE88] Orejas, F., Nivela, P., Ehrig, H. Categorical constructions for behavioural specifications, Dept. de LSI, Univ. Polit. de Catalunya, Barcelona 1988.

[Rei80] Reichel, H. Initially restricting algebraic theories, Proc. MFCS 80, Springer LNCS 88 (1980), pp. 504-514.

[San81] Sannella, D. A new semantics for Clear, Report CSR - 79 - 8, Univ. of Edinburgh, 1981.

[ST87a] Sannella, D; Tarlecki A. On observational equivalence and algebraic specification. J. Comp. and Sys. Sciences 34, pp. 150-178 (1987).

[ST87b] Sannella, D; Tarlecki A. Toward formal development of programs from algebraic specifications: implementations revisited. Proc. Joint Conf. on Theory and Practice of Software Development, Pisa, Springer LNCS 249, pp. 96-110 (1987).

[SW83] Sannella, D; Wirsing, M. A kernel language for algebraic specification and implementation, Proc. FCT-83, Springer LNCS 158, pp. 413-427, 1983.

[WPPDB83] Wirsing, M., Pepper, P., Partsch, H., Dosch, W., Broy, M. On hierarchies of abstract data types, Acta Informatica 20 (1983), pp. 1-33.