PARAMETERIZED HORN CLAUSE SPECIFICATIONS: PROOF THEORY AND CORRECTNESS

M. Navarro
Informatika Fakultatea
Euskal-Herriko Unibertsitatea
San Sebastian, SPAIN

F. Orejas
Facultat d'Informàtica
Universitat Politècnica de Catalunya
Barcelona, SPAIN

Recently, "algebraic" equational Horn clause specifications (or, in some sense, conditional specifications) have been advocated by several authors as the solution to some of the problems of Prolog [see, for instance, 11]. Most of the work done in this field has been dealing only with the operational aspects of such specifications (e.g. rewriting, narrowing, etc.), perhaps assuming that other kind of results will be direct generalizations of those obtained for the equational case.

However, there is an aspect that hinders, in many cases, this generalization: working with a (so-called) boolean constraint, i.e. having as admissible models for specifications algebras satisfying that a boolean sort contains only two values: true and false. Specifically, constructions that are almost trivial in the standard framework have to be approached with new techniques.

In this paper we study two aspects of parameterized specifications, proof theory and correctness. We characterize the inductive theory of a parameterized specification generalizing some results obtained by P. Padawitz in [15] (in particular some restrictions have been removed, for instance the need to have equality operators explicitly defined for every sort, or the need for persistency: we only ask for "bool-persistency"). Then, we obtain a proof theoretical characterization of three conditions related to the correctness of a parameterized specification: bool-persistency, (i.e. the property that ensures that the booleans are not "destroyed" by the parameterization), persistency (i.e. protection of the actual parameter) and passing compatibility (i.e. the property that assures the compatibility of the functorial and pushout semantics for parameter passing).

Other previous work related with our results is [8,5,16,14]. In [8] Ganzinger obtained the proof-theoretical characterization of persistency for the equational case. The characterizations of bool-persistency and persistency presented here are strongly inspired in his, indeed, the only-if part of our proofs is a direct generalization of his, but the if part presented the kind of problems mentioned above.

In [5] Ehrig dealt with parameterized specifications with arbitrary constraints (thus his work is more general), some of his results have been used in this paper, however his approach was model-theoretical due to the generality of his framework.

In [16] Padawitz obtained conditions for checking persistency of parameterized equational specifications with a boolean constraint. Although the similarity of the framework, the results are quite different, he was mainly involved in obtaining sufficient conditions for persistency that were easily checkable using rewriting techniques.

With respect to [14], the characterization of passing compatibility presented here is a straightforward generalization of the one presented there, once the new techniques used in the previous results are applied.

The organization of this paper is as follows: In section 1, we introduce briefly the basic concepts. In section two, we characterize the inductive theory defined by a parameterized specification. Finally, in section 3 we obtain the characterization of bool-persistency, persistency and passing compatibility.

ACKNOWLEDGEMENTS

The authors would like to thank P. Padawitz for showing us in [15] the use of the Ultrafilter Theorem. This work has been partially supported by Comisión Asesora de Investigación (ref. 2704-83)

1. Preliminaries

Familiarity with the usual notions concerning (parameterized) algebraic specifications is assumed (for detail, see [7]).

Given a set of sorts S, an S-sorted signature Σ is an indexed family of sets of operation symbols, $\Sigma = \{\Sigma_{w,s}\}_{w^* \in S, s \in S}$.

A Σ-algebra A consists of a family of sets (carriers or data domains) $\{A_s\}_{s \in S}$, and a family of operations

σ_A:A_{s1}x...xA_{sn}--->A_s for every σ in $\Sigma_{s1...sn,s}$ A <u>Σ-homomorphism</u> h: A ---> B, where A and B are Σ-algebras is a family of functions $\{h_s: A_s \text{---> } B_s\}_{s\in S}$ which commute with the operations. Σ-algebras together with their homomorphisms form the category Alg_Σ, having as initial object (up to isomorphism) the <u>term algebra</u> T_Σ. $T_\Sigma(X)$ stands for the <u>algebra of terms with variables in X,</u> i.e. the free Σ-algebra generated by X. Given an assignment a: X ---> A, there is a unique Σ-homomorphism a: $T_\Sigma(X)$ ---> A, extending a.

A Σ-algebra A <u>satisfies a (conditional) equation</u>,A |= λX.t=t' **if** t1=t1' & ... & tn=tn', with t,t',t1,t1',...,tn,tn' in $T_\Sigma(X)$, iff for every assignment a: X ---> A, if for every i ($1 \le i \le n$) a(ti)=a(ti') then a(t)=a(t'). A <u>satisfies a set of equations</u> E iff it satisfies every equation in E.

A <u>specification</u> SP is a triple (S, Σ ,E) formed by a set of sorts, a signature and a set of (conditional) equations.

Given a specification SP = (S,Σ,E), a Σ-algebra satisfying E is called a <u>SP-algebra</u>. SP-algebras together with their homomorphisms form the category Alg_{SP} with initial object $T_{SP} = T_\Sigma/{\equiv_E}$, where $\equiv_E$ stands for the congruence generated by E.

Given a specification SP = (S,Σ,E), <u>a combination</u> of SP and SP0 = (S0,Σ0,E0), denoted SP+SP0, is defined:

SP+SP0 = (S+S0,Σ+Σ0,E+E0)

where + denotes disjoint union. Note that SP0 does not need to be a specification (for instance, there may be a σ in $\Sigma0_{w,s}$ with ws in $(S+S0)^+ - S0^+$, but SP+SP0 does.

A <u>specification morphism</u> h: SP1 ---> SP2 consists of a function h:S1--->S2 and a family of functions $\{h_{w,s}:\Sigma1_{w,s} \text{--->} \Sigma2_{h^*(w),h(s)}\}_{w^*\in S, s\in S}$ (where h*(s1...sn) denotes h(s1)...h(sn)), such that E2 ⊇ h(E1), i.e. every equation in E1 when translated through h belongs to E2. Specifications together with their morphisms form the category CATSP.

Every specification morphism h: SP1 ---> SP2 induces a functor U_h: Alg_{SP2}---> Alg_{SP1} called the <u>forgetful functor</u> associated to h, defined U_h(A2)=A1 iff

$\forall s\in S1 \qquad A1_s = A2_{h(s)}$

$\forall\sigma\in\Sigma1_{w,s}\ \ \sigma_{A1} = (h_{w,s}(\sigma))_{A2}$

U_h has a left adjoint F_h: Alg_{SP1} ---> Alg_{SP2}, called the free functor associated to h.

From now on, we shall assume that every specification contains, as a subspecification, the boolean specification. Also, we will not allow non boolean operations having boolean parameters, i.e. if $\sigma\in\Sigma_{\omega,bool}-\Sigma BOOL$, then $w\in(S\text{-}\{bool\})^*$. That is, we are considering booleans as special values: we may define boolean-valued functions (predicates) but they may not be parameters.

Moreover, we shall assume that equations take the form $\lambda X.t=t'$ **if** C where C is a $\Sigma(X)$-condition, i.e. a $\Sigma(X)$-term of boolean sort. Though the abuse of notation, conditions may denote, as above, boolean sorted equations of the kind: C=true. Equations of the kind:

$\lambda X.t=t'$ **if** true

will often be abreviated to:

$\lambda X.t=t'$

Given a specification SP, the category LOGALG(SP) shall denote the full subcategory of Alg_{SP}, whose objects are algebras A satisfying that $U_{bool}(A) = \mathbf{B}$ (where bool is the inclusion morphism from the boolean specification BOOL to SP and **B** is the boolean algebra of two elements).

In [13] two proof systems, |- and $|\text{-}_L$, were given satisfying:

SP |- $\lambda X.t=t'$ **if** C iff $\forall A\in Alg_{SP}$ A |= $\lambda X.t=t'$ **if** C

SP $|\text{-}_L$ $\lambda X.t=t'$ **if** C iff $\forall A\in$ LOGALG(SP) A |= $\lambda X.t=t'$ **if** C

|- is just a generalization to the many sorted case of a proof system given by Selman in [17] using the technique devised by Goguen and Meseguer in [10] to deal with many-sorts. $|\text{-}_L$ is an adaptation of another proof system given by Selman in the same paper adding rules to cope with the boolean constraint.

Note that SP |- $\lambda X.t=t'$ implies SP $|\text{-}_L$ $\lambda X.t=t'$ but the converse is not true, even if the terms t1 and t2 contain no variables. For instance, if SP contains the equations:

$\lambda X.t=t'$ **if** C

$\lambda X.t=t'$ **if** not(C)

then SP $\vdash_L$ $\lambda X.t=t'$ but not necessarily SP $\vdash$ $\lambda X.t=t'$.

A set of conditions COND is <u>non contradicting</u> with respect to a set of equations E iff

$$E+COND^* \not\vdash_L \text{true=false}$$

where COND* is the same as COND, but considering its variables as constants. From now on, although the abuse of notation and if there is no possible confusion, we will not distinguish between COND and COND*.

A <u>parameterized data type</u> PDT is a triple (PAR,BODY,H), where PAR = (SPAR,ΣPAR,EPAR) is the <u>parameter declaration</u>, BODY = (SBODY,ΣBODY,EBODY) = PAR + (S2,Σ2,E2) is called the <u>target specification</u> and H is a functor, H: LOGALG(PAR) ---> LOGALG(BODY) (we assume H equipped with a natural family of homomorphisms I_A: A ---> U_i(H(A)), where i is the inclusion morphism from PAR to BODY). H is <u>persistent (strongly persistent)</u> iff for every A in LOGALG(PAR), I_A is an isomorphism (the identity).

A <u>parameterized specification</u> PSP is a pair (PAR,BODY), where PAR and BODY are as in the previous definition and satisfy <u>bool-persistency</u>, i.e. for every A in LOGALG(PAR), $U_{bool}(F_i(A))=$ **B**, where F_i is the free functor associated to the inclusion morphism from PAR to BODY. The semantics of PSP is considered to be the parameterized data type (PAR,BODY,F_i). We shall say that PSP is persistent if F_i is persistent or strongly persistent.

Often, parameterized (conditional) specifications are not persistent if we consider as admissible parameter any PAR-algebra, although they are persistent when we do restrict to LOGALG(PAR). This happens with the following example:

Example 1.1

Let PAR be the following specification:

PAR = BOOL + **sorts** data
ops eq: data x data ---> bool

eqns 1) $\lambda x.\ eq(x,x) = true$

2) $\lambda\{x,y\}.x=y$ **if** $eq(x,y)$

and let BODY be:

BODY = PAR + **sorts** set

ops empty: set

insert: set x data ---> set

is_in: set x data ---> bool

eqns 3) $\lambda\{s,x,y\}.\ insert(insert(s,x),y)=insert(insert(s,y),x)$

4) $\lambda\{s,x\}.\ insert(insert(s,x),x)= insert(s,x)$

5) $\lambda x.is\_in(empty,x)=false$

6) $\lambda\{s,x\}.is\_in(insert(s,x),x)=true$

7) $\lambda\{s,x,y\}.is\_in(insert(s,x),y)=is\_in(s,y)$ **if** $not(eq(x,y))$

This parameterization, as we shall see later, works perfectly well (it is persistent) if we restrict admissible parameters to those in LOGALG(PAR), i.e. those in which the boolean values are {true,false} and eq is equality, but is not persistent (it may add "junk" to the parameter) if we do not restrict the class of admissible parameters. Changing the specification (for instance, adding more equations) would not help to solve the problem. ♦

Now, we may define standard parameter passing at the specification level: given a parameterized specification PSP = (PAR,BODY), with PAR = (SPAR,ΣPAR,EPAR) and BODY = (SBODY,ΣBODY,EBODY) = PAR + (S2,Σ2,E2), a specification ACT = (SACT,ΣACT,EACT) called actual parameter specification and a morphism h1: PAR ---> ACT, called parameter passing morphism, the mechanism of parameter passing may be described by the following pushout diagram:

$$\begin{tikzcd}
PAR \arrow[r, "i1"] \arrow[d, "h1"'] & BODY \arrow[d, "h2"] \\
ACT \arrow[r, "i2"'] & VAL
\end{tikzcd}$$

where i1 is the inclusion morphism. VAL is called the value specification. More concretely, VAL = (SVAL,ΣVAL,EVAL) = ACT + (S4,Σ4,E4), with S4 = S2, Σ4 = h2(Σ2) and E4 = h2(E2), i2 is the inclusion morphism and h2 is defined:

$$h2(s) = \text{if } s \in S2 \text{ then } s \text{ else } h1(s)$$

$$h2_{w,s}(\sigma) = \text{if } \sigma \in \Sigma 2 \text{ then } \sigma \text{ else } h1_{w,s}(\sigma)$$

Parameter passing is correct iff the following two conditions hold, for every A in LOGALG(ACT):

1) Actual parameter protection: $U_{i2}(F_{i2}(A)) = A$

2) Passing compatibility: $F_{i1}(U_{h1}(A) = U_{h2}(F_{i2}(A))$

A parameterized specification is correct (resp. satisfies passing compatibility) if for all possible actual parameter specifications (and parameter passing morphisms) parameter passing is correct (resp. satisfies passing compatibility). In [5] it is proved that PSP is correct iff it is persistent.

2. The inductive theory of a parameterized specification

Given a specification SP, the theory defined by this specification consists of all the equations deducible from SP, which (if the proof system is sound and complete) coincide with the set of equations satisfied by all models of SP.

However, often we are not interested in all models satisfying SP. For instance if the specification is not parameterized we may be interested only in finitely generated models, or if it is parameterized on models finitely generated from the actual parameter. The set of equations satisfied by all models finitely generated (from the actual parameter) satisfying a (parameterized) specification is called the inductive theory defined by the specification:

Definition 2.1

Given a parameterized specification PSP=(PAR,BODY), we define the inductive equational theory defined by PSP:

$$IND(PSP) = \{ \lambda X.t=t' / \forall A \in LOGALG(PAR)\ F(A) \models \lambda X.t=t' \}$$

In Theorem 2.4 we will characterize IND(PSP) in terms of the (non-conditional) equations satisfied by certain free algebras, but before that we have to see two lemmas:

NOTE From now on, given two $\Sigma(X)$-terms t1, t2 and a $\Sigma(X)$-condition C, we shall say that $E|\!-_L$ t1=t2 **if** C (instead of $E|\!-_L$ λX.t1=t2 **if** C) if from E we may deduce this equation considering the variables as constants, i.e. considering t1 and t2 as ground terms and C as a ground condition.

Lemma 2.2

Given SP = (S,Σ,E) and a set of $\Sigma(X)$-conditions COND such that COND is non contradicting w.r.t. E, then there is a set of bool-sorted equations E(COND) such that $T_\Sigma(X)/\equiv_{E+E(COND)}$ satisfies every condition in COND and belongs to LOGALG(SP).

Proof

Let A = $T_\Sigma(X)/\equiv_{Der(E)}$, where Der(E) denotes the set of equations t=t' such that E $|\!-_L$ λX.t=t'. Obviously, $U_{Bool}(A)$ is a boolean algebra. Let COND' be COND **U** {not(C)/ SP$|\!-_L$ true=false **if** C}, COND' denotes a set of values in $U_{Bool}(A)$ satisfying the finite intersection property (i.e. the conjunction of any finite subset of boolean values is not equal to false) since COND is non contradicting w.r.t. E, then, according to a corollary of the Ultrafilter Theorem (cf. [2]). there is an ultrafilter U containing all the values denoted by COND'. Finally, we define E(COND) as {t1=true/ t1 denote a value inside U} **U** {t1=false/ t1 denote a value outside U}. By construction, $T_\Sigma(X)/\equiv_{E+E(COND)}$ satisfies every C in COND and belongs to LOGALG(SP) since, on one hand, by construction in $U_{bool}(T_\Sigma(X)/\equiv_{E+E(COND)})$ there will be at most two elements, true and false, and, on the other hand they are different because it may be proved that for any pair of boolean sorted terms, t1 and t2 in $T_\Sigma(X)$, $T_\Sigma(X)/\equiv_{E+E(COND)}|=$ t1=t2 iff there is a ΣPAR(X)-condition C such that $E|\!-_L$ λX.t1=t2 **if** C and <C=true>$\in$E(COND). But if $E|\!-_L$ λX.true=false **if** C then, by construction, only <not(C)=true> and <C=false> belong to E(COND). ♦

Lemma 2.3

Given SP = (Σ,E), a set of $\Sigma(X)$-conditions COND and two $\Sigma(X)$-terms t1 and t2 such that SP+COND*$\not|\!-_L$ t1=t2, there is a set of bool-sorted equations E(COND,t1,t2) such that A=$T_\Sigma(X)/\equiv_{E+E(COND,t1,t2)}$ belongs to LOGALG(SP), A $|\!\neq$t1=t2 and A satisfies every C in COND.

Proof

Let COND' be COND **U** {not(C)/ SP$|\!-_L$t1=t2 if C}, by assumption COND' is not contradicting w.r.t. SP, thus applying lemma 2.2, there is a set of equations E(COND') such that $A=T_{\Sigma}(X)/\equiv_{E+E(COND')}$ satisfies every C in COND' (and, thus, in COND) and belongs to LOGALG(SP). On the other hand $A \not\models t1=t2$, since otherwise a condition in COND' would be false in A, contradicting one of the two previous statements. ♦

Theorem 2.4

Let PSP be the parameterized specification (PAR,BODY) and let X_{PAR} be an (SPAR-{bool})-sorted denumerable set of variables then:

$$\lambda X.t1=t2 \in IND(PSP) \text{ iff } T_{\Sigma BODY}(X_{PAR})/\equiv_{Der(EBODY)} \models \lambda X.t1=t2.$$

Proof

=>) Suppose $T_{\Sigma BODY}(X_{PAR})/\equiv_{Der(EBODY)} \not\models a(t1)=a(t2)$ for a given assignment a: X-->, $T_{\Sigma BODY}(X_{PAR})$ then, according to the previous lemma there is a set of bool-sorted equations E(∅,a(t1),a(t2)) such that $A=T_{\Sigma BODY}(X_{PAR})/\equiv_{E+E(\varnothing,a(t1),a(t2))}$ belongs to LOGALG(BODY) and $A \not\models a(t1)=a(t2)$. Let U(A) be the PAR-algebra obtained after applying the forgetful functor to A. Obviously, U(A) is in LOGALG(PAR), moreover $F(U(A)) \not\models a(t1)=a(t2)$ since otherwise a(t1) and a(t2) would be equal in A. Note, however, that we do not need F to be persistent (although, it is assumed to be bool-persistent).

<=) If $T_{\Sigma BODY}(X_{PAR})/\equiv_{Der(E)} \models t1=t2$, then for every algebra A in LOGALG(PAR) and every assignment h: X ---> F(A) there is a (unique) homomorphism h^*: $T_{\Sigma BODY}(X_{PAR})/\equiv_{Der(E)}$ ---> F(A), thus $F(A) \models t1=t2$. ♦

3. Correctness of parameterized specifications

As we have seen in the preliminaries, three conditions are asked for the correctness of parameterized specifications: bool-persistency, actual parameter protection (persistency [5]), and passing compatibility.

In this section we are going to characterize proof-theoretically bool-persistency (Theorem 3.1) and persistency (Theorem 3.3) in terms of consistency and completeness conditions. After, we will characterize passing compatibility in terms of persistency (Theorem 3.4).

Theorem 3.1

PSP = (PAR,BODY) is bool-persistent iff PSP satisfies the following two properties:

1. Bool-consistency: BODY $\not\vdash_L$ true=false

2. <u>Bool-completeness</u>: For every t in $T_{\Sigma BODY}(X_{PAR})$ of boolean sort there are t1,...,tn in $T_{\Sigma PAR}(X_{PAR})$ and $\Sigma PAR(X)$-conditions C1,..., Cn, such that BODY$|\text{-}_L$ λX.t=ti if Ci (for every i, $1 \le i \le n$), and PAR$|\text{-}_L$ λX.C1$\vee ... \vee$Cn = true.

Proof

=>) If PSP is not bool-consistent, obviously, PSP is not persistent w.r.t booleans. Assume PSP is not bool-complete, let t be the boolean sorted ΣBODY(X)-term for which there is not a finite set of terms t1,...,tn and ΣPAR(X)-conditions C1, ..., Cn such that BODY$|\text{-}_L$ λX.t=ti if Ci (for every i, $1 \le i \le n$), and PAR $|\text{-}_L$ λX.C1$\vee ... \vee$Cn = true. Let COND be {not(C)/C is a ΣPAR(X)-condition and $\exists$t' in $T_{\Sigma PAR}(X)$ PSP $|\text{-}_L$ λX.t=t' if C}, by assumption COND is non contradicting w.r.t EPAR thus according to lemma 2.2 A = $T_{\Sigma PAR}(X)/\equiv_{EPAR+E(COND)}$ is in LOGALG(PAR) and every not(C) in COND is true in A (i.e. every C is false in A). Clearly, in F(A) t is not congruent neither with true nor with false, since otherwise a condition in COND would be false in F(A).

<=) Let A be in LOGALG(PAR), for every t in $T_{\Sigma BODY}(A)_{bool}$, since A is a LOGALG(PAR)-algebra and SP is bool-complete, there is a t_i in $T_{\Sigma PAR}(A)$ and a ΣPAR(A)-condition C_i such that:

(*) SP $|\text{-}_L$ $t=t_i$ if C_i and A$|$= C_i=true.

This implies F(A)$|$= $t=t_i$.

Assume F(A)$|$= t=t', with t,t' in $T_{\Sigma PAR}(A)$, this means that there is a sequence of terms $t_1,..., t_n$ such that $t=t_1$, $t'=t_n$, and for every i ($1 \le i < n$) $t_i \leftarrow\rightarrow_{BODY+EA} t_{i+1}$, we will define a sequence of ΣPAR(A)-terms $t_1',...,t_n'$, (with $t_1'=t_1$ and $t_n'=t_n$) and of ΣPAR(A)-conditions $C_1, ...,C_n$ such that for every m ($0 \le m < n$):

a) BODY $|\text{-}_L$ $t_m=t_m'$ if Cm

b) EA $|\text{-}_L$ C_m=true

c) EA $|\text{-}_L$ $t_m'=t_{m+1}'$

It should be clear that if such sequences of terms and ΣPAR(A)-conditions exist then A$|$= t=t'.

In the definition of t_{i+1}' and C_{i+1} we have two cases:

case 1: BODY $|\text{-}_L\ t_i=t_{i+1}$ **if** C_i' and BODY+EA $|\text{-}_L\ C_i'$=true. By bool completeness, there are ΣPAR(A)-conditions $C_{i1},C_{i1}',\dots,C_{ik},C_{ik}'$, such that BODY $|\text{-}_L\ C_i'=C_{ij}$ **if** C_{ij}' for every j $(1\leq j\leq k)$ and PAR $|\text{-}_L\ C_{i1}' \vee \dots \vee C_{ik}'$=true. This means that there is a j such that BODY $|\text{-}_L\ t_i=t_{i+1}$ **if** $C_{ij}\&C_{ij}'$ and A $|= C_{ij}\&C_{ij}'$=true. Now, using (*) above, there is a ΣPAR(A)-term t_{i+1}' and a ΣPAR(A)-condition C_{i+1} such that: BODY $|\text{-}_L\ t_{i+1}=t_{i+1}'$ **if** C_{i+1} and A$|= C_{i+1}$=true. By construction conditions a) and b) hold trivially, let us see that c) also holds: By induction, we know that BODY $|\text{-}_L\ t_i=t_i'$ **if** C_i, we also have BODY $|\text{-}_L\ t_i=t_{i+1}$ **if** $C_{ij}\ \&\ C_{ij}'$ and BODY $|\text{-}_L\ t_{i+1}=t_{i+1}'$ **if** C_{i+1} hence, by transitivity, BODY $|\text{-}_L\ t_i'=t_{i+1}'$ **if** $C_i\&C_{ij}\&C_{ij}'\&C_{i+1}$, but, by consistency, this means that PAR $|\text{-}_L\ t_i'=t_{i+1}'$ **if** $C_i\&C_{ij}\&C_{ij}'\&C_{i+1}$, that is A$|= t_i'=t_{i+1}'$ since C_i,C_{ij},C_{ij}' and C_{i+1} are true in A.

case 2: EA $|\text{-}_L\ t_i=t_{i+1}$. This means that there are terms $l,r \in T_{\Sigma PAR}(A)$ and $t \in T_{\Sigma BODY}(A\mathbf{U}\{x\})$ such that <l=r>$\in$EA, f1(t)=t_i and f2(t)=t_{i+1}, where f1 and f2 substitute, respectively, x by l and x by r. Now, by sufficient completeness, there is a ΣPAR(A**U**{x})-term t' and a ΣPAR(A**U**{x})-condition C' such that BODY $|\text{-}_L$ t=t' **if** C' and A$|=$ f1(C'). Let t_{i+1}' and C_{i+1} be, respectively, f2(t') and f2(C'), obviously BODY $|\text{-}_L\ t_{i+1}=t_{i+1}'$ **if** C_{i+1} and A$|= C_{i+1}$=true, moreover, A$|= t_i'=t_{i+1}'$ since, by transitivity, BODY+EA $|\text{-}_L\ t_i'=t_{i+1}'$ **if** $C_i\&C_{i+1}$, and thus, by consistency, PAR+EA $|\text{-}_L\ t_i'=t_{i+1}'$ **if** $C_i\&C_{i+1}$. ♦

Example 3.2
It should be clear that the specification of example 1.1 is bool-consistent, let us see that it is also bool-complete.

Every term t in $T_{\Sigma BODY}(X_{PAR})_{bool} - T_{\Sigma PAR}(X_{PAR})$ is of the form: is_in(insert(....(insert(empty,x1),...),xn),y). We will proceed by induction:

case n=0 Trivial: BODY $|\text{-}_L\ \lambda x$.is_in(empty,x)=false

case n=k+1 On one hand we have:

BODY $|\text{-}_L\ \lambda\{s,x,y\}$.is_in(insert(s,x),y)=is_in(s,y) **if** not(eq(x,y))

On the other using equation 2) and substitutivity:

BODY $|\text{-}_L\ \lambda\{s,x,y\}$.is_in(insert(s,x),y)= is_in(insert(s,x),x) **if** eq(x,y) and by equation 6) and transitivity:

BODY $|\!-_L$ λ\{s,x,y\}.is_in(insert(s,x),y)=true if eq(x,y)

Finally, trivially:

PAR $|\!-_L$ λ\{x,y\} eq(x,y) v not(eq(x,y)= true ◆

Theorem 3.3

PSP = (PAR,BODY) is persistent in LOGALG(PAR) iff PSP satisfies the following two properties:

1. Consistency: For every t1,t2 in $T_{\Sigma PAR}(X)$ and every ΣPAR(X)-condition C we have PAR $|\!-_L$ λX.t1=t2 if C iff BODY $|\!-_L$ λX.t1=t2 **if** C.

2. Sufficient completeness: For every t in $T_{\Sigma BODY}(X_{PAR})$ of sort in PAR, there are t1,...,tn in $T_{\Sigma PAR}(X_{PAR})$ and ΣPAR(X)-conditions C1,..., Cn, such that BODY $|\!-_L$ λX.t=ti **if** Ci (for every i, 1≤i≤n), and PAR $|\!-_L$ λX.C1 v ... v Cn = true.

Proof

=>) Assume PSP is not consistent, i.e. there are terms t1, t2 and a ΣPAR(X)-condition C such that BODY $|\!-_L$ λX.t1=t2 **if** C and PAR $\not\vdash_L$ λX.t1=t2 **if** C. Obviously PAR+C $\not\vdash_L$ t1=t2, since otherwise λX.t1=t2 **if** C would be trivially deducible from PAR. Hence, according to lemma 2.3 $T_{\Sigma PAR}(X)/\equiv_{EPAR+E(C,t1,t2)}$ is in LOGALG(PAR), C is true in A and in A $|\neq$ t1=t2. On the other hand, obviously, in F(A) $|=$ t1=t2.

Assume PSP is not sufficiently complete, let t be the ΣBODY(X)-term for which there is not a finite set of terms t1,...,tn and ΣPAR(X)-conditions C1, ..., Cn such that BODY $|\!-_L$ λX.t=ti **if** Ci (for every i, 1≤i≤n), and PAR $|\!-_L$ λX.C1 v ... v Cn = true. Let COND be \{not(C)/ C is a ΣPAR(X)-condition and $\exists$t' in $T_{\Sigma PAR}(X)$ PSP $|\!-_L$ λX.t=t' **if** C\}, by assumption COND is non contradicting w.r.t EPAR thus according to lemma 2.2 A = $T_{\Sigma PAR}(X)/\equiv_{EPAR+E(COND)}$ is in LOGALG(PAR) and every not(C) in COND is true in A (i.e. every C is false in A). Now, in F(A) t is not congruent to any value of A.

<=) Similar to the same part of theorem 3.1. ◆

In [14] it was proved that for the equational case passing compatibility was almost persistency (persistency or trivial inconsistency), here, using similar techniques, we are going to prove that persistency is exactly passing compatibility. The reason is that we are assuming bool-persistency and, thus, avoiding trivial inconsistency.

Theorem 3.4

PSP satisfies passing compatibility for every logical parameter iff PSP is persistent.

Proof

=>) Assume PSP is not consistent (but remeber that PSP is assumed to be bool-persistent), then there are two $\Sigma PAR(X)_s$-terms t1 and t2 and a $\Sigma PAR(X)$-condition C such that PAR $\not\vdash_L$ $\lambda X.t1=t2$ **if** C and BODY $\vdash_L$ $\lambda X.t1=t2$ **if** C. Let SP' be the specification PAR+($\varnothing$,Σ',E'), where Σ' consists of X (taken as constants of appropriate sorts) plus an operation c: s --> bool, and E´ consists of the equations:

$$c(t1) = true$$
$$c(t2) = false$$

Clearly, C is non contradicting w.r.t. EPAR+E', then, according to Lemma 2.2, there is a set of equations E(C) such that A = $T_{\Sigma PAR+\Sigma'}/_{\equiv EPAR+E'+E(C)}$ is in LOGALG(SP), A|= C=true and A|$\neq$ t1=t2 (otherwise A would not be in LOGALG(SP').

Now,let ACT be SP'+($\varnothing$, $\varnothing$,E(C)), let the parameter passing morphism h1 be the inclusion morphism, then in $F_{i2}(A)$ true is equal to false, but not in $F_{i1}(U_{h1}(A))$.

Assume PSP is not sufficiently complete, let t be the $\Sigma BODY(X)$-term for which there is not a finite set of terms t1,...,tn and $\Sigma PAR(X)$-conditions C1, ..., Cn such that BODY $\vdash_L$ $\lambda X.t=ti$ **if** Ci (for every i, $1 \le i \le n$), and PAR $\vdash_L$ $\lambda X.C1 \vee ... \vee Cn = true$. Let SP' be the specification PAR+(SPAR',Σ',E'), where SPAR' is a copy of SPAR excluding bool (i.e. SPAR'= {s'/ s$\in$SPAR-{bool}}), Σ' consists of X (taken as constants of appropriate sorts) plus two operations c_s: s --> s' and u_s: s' --> s, for every s in SPAR-{bool}, and E´ consists of the equations:

$$u_s c_s(t) = t$$

for every s in SPAR-{bool} and every t in $T_{\Sigma PAR+\Sigma'}$. Now, let COND be {not(C)/ C is a $\Sigma PAR(X)$-condition and $\exists t'$ in $T_{\Sigma PAR}(X)$ PSP $|-_L \lambda X.t=t'$ if C}, COND is non contradicting w.r.t EPAR+E' thus according to lemma 2.2 $A = T_{\Sigma PAR+\Sigma'}/_{\equiv EPAR+E'+E(COND)}$ is in LOGALG(SP') and every not(C) in COND is true in A (i.e. every C is false in A).

Let ACT=SP'+(∅,∅,E(COND)), let the parameter passing morphism h1 be the inclusion morphism, then $F_{i1}(U_{h1}(T_{ACT})) \neq U_{h2}(F_{i2}(T_{ACT}))$. The reason is the following: F_{i1} generates some junk on $U_{h1}(T_{ACT})$ (at least the term t would be junk, if we consider its variables as constant symbols from Σ'), but on $U_{h2}(F_{i2}(T_{ACT}))$ we have generated, at least, the double of junk: for every junk element t of sort s generated by F_{i1}, in $F_{i2}(T_{ACT})$ we have the same element plus $u_s c_s(t)$.

<=) See [5] ♦

4. References

[1] Arbib, M.E.; Manes, E.G.: "Arrows, structures and functors: the categorical imperative", Academic Press 1975.

[2] Bell, J.L.; Slomson, A.B.: "Models and Ultraproducts: an Introduction", North-Holland (1971)

[3] 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.

[4] Ehrich, H.-D.: "On the theory of specification, implementation and parameterization of abstract data types", JACM 29,1 (1982), pp. 206-227.

[5] Ehrig, H.: "Algebraic theory of parameterized specifications with requirements", Proc. 6th. CAAP, Springer LNCS 112, pp. 1-24, 1981.

[6] Ehrig, H.; Kreowski, H.-J.; Thatcher, J.W.; Wagner, E.G.; Wright, J.B.: "Parameter passing in algebraic specification languages", Proc. Aarhus Workshop on Program Specification, Springer LNCS 134, 1981.

[7] Ehrig, H.; Mahr, B.: "Fundamentals of algebraic specification 1", Springer EATCS Monographs on Theor. Comp. Sc., 1985.

[8] Ganzinger, H.: "Parameterized specifications: parameter passing and implementation with respect

to observability", TOPLAS 5,3 (1983), pp. 318-354.

[9] Goguen, J.A.; Meseguer, J: "Universal realization, persistent interconnection and implementation of abstract modules", Proc. IX ICALP, Springer LNCS 140, pp. 265-281, 1982.

[10] Goguen, J.A.; Meseguer, J: "Completeness of many-sorted equational logic", Sigplan Notices 16,7 (1981) pp 24-32.

[11] Goguen, J.A.; Meseguer, J: "Equality, types, modules and (why not?) generics for logic programming", The Journal of Logic Programming 1,2 (1984), pp. 179-210.

[12] Goguen, J.A.; Thatcher, J.W.; Wagner, E.G.: "An initial algebra approach to the specification, correctness and implementation of abstract data types", in 'Current Trends in Programming Methodology, Vol IV: Data Structuring', R.T. Yeh (ed.), Prentice Hall 1978, pp. 80-149.

[13] Navarro, M.; Orejas, F.: "Proof rules for conditional equations", Res. Rep., Facultat d'Informatica de Barcelona, 1986.

[14] Orejas, F.: "Passing compatibility is almost persistency", in 'Recent trends on data type specification' H.-J. Kreowski (ed.)Springer IFB 114, 1985.

[15] Padawitz, P. : "Towards a proof theory of parameterized specifications", in 'Semantics of Data Types', G. Kahn, D.B. MacQueen, G. Plotkin (eds.), Springer LNCS 173 (1984), pp. 375-391.

[16] Padawitz, P. : "Parameter preserving data type specifications", in 'Formal Methods and Software Development, vol I', H. Ehrig, Ch. Floyd (eds.) Springer LNCS 186 (1985), pp. 323-341.

[17] Selman, A. : "Completeness of calculii for axiomatically defined classes of algebras", Algebra Universalis, 2, 1 (1972), pp. 20-32.

[18] Thatcher, J.W.; Wagner, E.G.; Wright, J.B.: "Data type specification: parameterization and the power of specification techniques", Proc. 10th STOC, San Diego, Ca., 1978.