Proofs in parameterized specifications

Hélène Kirchner, Claude Kirchner

UNITÉ DE RECHERCHE
INRIA-LORRAINE

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P. 105
78153 Le Chesnay Cedex
France
Tél.:(1) 39 63 55 11

Rapports de Recherche

N° 1424

Programme 2
*Calcul Symbolique, Programmation
et Génie logiciel*

PROOFS IN PARAMETERIZED SPECIFICATIONS

Hélène KIRCHNER

Avril 1991

* R R _ 1 4 2 4 *

Proofs in Parameterized Specifications

Hélène Kirchner
CRIN-CNRS & INRIA-Lorraine
BP 239
54506 Vandœuvre-lès-Nancy Cedex
France
e-mail: hkirchner@loria.crin.fr

Abstract

Theorem proving in parameterized specifications has strong connections with inductive theorem proving. An equational theorem holds in the generic theory of the parameterized specification if and only if it holds in the so-called generic algebra. Provided persistency, for any specification morphism, the translated equality holds in the initial algebra of the instantiated specification. Using a notion of generic ground reducibility, a persistency proof can be reduced to a proof of a protected enrichment. Effective tools for these proofs are studied in this paper.

Preuves dans les spécifications paramétrées

Hélène Kirchner
CRIN-CNRS & INRIA-Lorraine
BP 239
54506 Vandœuvre-lès-Nancy Cedex
France
e-mail: hkirchner@loria.crin.fr

Abstract

La preuve de théorèmes dans les spécifications paramétrées a de fortes relations avec les preuves par récurrence dans une algèbre initiale. Un théorème égalitaire est valide dans la théorie générique de la spécification paramétrée si et seulement si il est valide dans une algèbre dite "générique". En supposant la propriété de persistence, un tel théorème est valide dans l'algèbre initiale de toute spécification instanciée, modulo le morphisme de spécification. En utilisant une notion de réductibilité close générique, une preuve de persistence se ramène à une preuve de protection d'enrichissement. Des outils effectifs sont développés pour ce type de preuves.

1 Introduction

Parameterization is a generic way for building families of specifications and for reusing specifications. An important concern is to also make use of parameterization at the proof level and to develop a generic proof method. As argued in [7], this approach has several advantages: First, it allows performing proofs in a structured way that reflects the program structure. So generic proofs are performed for parameterized equational specifications. Second, a generic proof for a parameterized specification must be given only once and can be reused for each instantiation of the parameter.

Rather than promoting completely new ideas on the subject of parameterization, this paper is aimed to clarify some points, to gather different ideas and combine several results presented in [2, 7, 8, 9, 15, 17, 24]. More precisely, the goals here are the following ones:

- to emphasize the connection between (protected) enrichment and (persistent) parameterized specifications.
- to show that a persistency proof can be reduced to a proof of a protected enrichment, using an adequate notion of generic ground reducibility.
- to precise the use of persistency for theorem proving: an equational theorem holds in the generic theory of the parameterized specification if and only if it is an inductive theorem in a particular initial algebra, called the generic algebra. Provided persistency, such a theorem is generic: for any specification morphism m, the translated equality using m holds in the initial algebra of the instantiated specification.
- to provide effective tools to prove that a parameterized specification is persistent, to prove generic theorems and to prove generic ground reducibility.

Sections 2 and 3 recall the necessary definitions and results about enrichment and parameterization. Section 4 defines the central notion of generic ground reducibility. Section 5 is devoted to provide tools for proving persistency, while Section 6 is concerned with generic theorem proving.

2 Enrichments

All notations are compatible with [14]. Given a many-sorted signature Σ, the set of terms built on Σ and a denumerable set of sorted variables $\mathcal{X}$ is denoted by $\mathcal{T}(\Sigma \cup \mathcal{X})$. $\mathcal{V}(t)$ denotes the set of variables occurring in the term t.

Definition 1 *A* specification, *denoted $SP = (\Sigma, E)$ is given by a signature Σ, composed of a set of* sort symbols *S and a set of* function symbols *$\mathcal{F}$ with rank declarations, and a set E of universally quantified equalities $(\forall X, t = t')$ where $\mathcal{V}(t) \cup \mathcal{V}(t') \subseteq X$. (The quantification may be omitted when $X = \mathcal{V}(t) \cup \mathcal{V}(t')$).*

A specification $SP = (\Sigma, E)$ actually describes a class of algebras, namely the class of Σ-algebras satisfying the equalities E, denoted $ALG(SP)$. $ALG(SP)$ with SP-homomorphisms is a category also denoted by $ALG(SP)$.

Componentwise inclusion of specifications corresponds to enrichments.

Definition 2 *An* enrichment *of a specification $SP = (\Sigma, E)$ is a specification $SP' = (\Sigma', E')$ such that $\Sigma \subseteq \Sigma'$ and $E \subseteq E'$.*

SP is often referred to as the *primitive* or *basic* specification, while SP' is called the *enriched* specification.

A forgetful functor is associated to an enrichment.

Definition 3 *Assume that $SP \subseteq SP'$. Then the* forgetful functor $\mathcal{V}$ *from $ALG(SP')$ to $ALG(SP)$ is defined as follows:*

- $\forall A' \in ALG(SP')$, $A = \mathcal{V}(A')$ *is the SP-algebra such that* $\forall s \in \Sigma, A_s = A'_s$ *and* $\forall f \in \Sigma, f_A = f_{A'}$.
- $\forall h'$ *SP'-morphism,* $h = \mathcal{V}(h')$ *is the SP-homomorphism such that* $h_s = h'_s$ *for any* $s \in \Sigma$.

Let $\overset{*}{\longleftrightarrow}_E$ denote the replacement of equals by equals on $\mathcal{T}(\Sigma \cup \mathcal{X})$, which is correct and complete for deduction in $ALG(SP)$:

$$t \overset{*}{\longleftrightarrow}_E t' \text{ iff } \forall \mathcal{A} \in ALG(SP), \mathcal{A} \models (\forall X, t = t').$$

Enrichments are classified according to their effect on the initial algebra of the enriched specification. Mainly, enrichments can produce *junks*, that is new terms that are not equivalent to an already existing term, or *confusions*, that is new equivalences between terms originally distincts.

Definition 4 *Let $SP = (\Sigma, E) \subseteq SP' = (\Sigma', E')$ be an enrichment.*

The enrichment is consistent *if for any sort $s \in \Sigma$, any ground terms t and t' of sort s in $\mathcal{T}(\Sigma)$, $t \overset{*}{\longleftrightarrow}_E t'$ iff $t \overset{*}{\longleftrightarrow}_{E'} t'$.*

The enrichment is sufficiently complete *if for any sort $s \in \Sigma$, any ground term t' of sort s in $\mathcal{T}(\Sigma')$, there exists a term t of sort s in $\mathcal{T}(\Sigma)$ such that $t \overset{*}{\longleftrightarrow}_{E'} t'$.*

An enrichment which is both consistent and sufficiently complete is said protected.

In the case where the theories are presented by ground convergent rewrite systems (i.e. confluent and terminating on ground terms), a proof by consistency method has been developped by [2] to prove theorems in initial algebras. The method is based on the notion of ground reducibility. Given a ground convergent rewrite system R, a term t is *ground reducible* with R if all its ground instances are R-reducible. An equality $(\forall X, t = t')$ is *ground reducible* with R if for any ground substitution σ such that $\sigma(t) \neq \sigma(t')$, either $\sigma(t)$ or $\sigma(t')$ is R-reducible. The property of ground reducibility is decidable for finite rewrite systems [17, 25]. Algorithms for deciding ground reducibility in the case of left-linear rules have been given, for instance in [15, 17, 4, 20, 22]. The general case is considered in [6, 5, 18]. Ground reducibility for a class rewrite system R/E is undecidable when E is a set of associative and commutative axioms [17] but is decidable when R is left-linear [16]. The sufficient completeness is in general undecidable [17] but in some cases, it is equivalent to ground reducibility.

The goal of this paper is to extend these proofs techniques to parameterized specifications.

3 Parameterization

3.1 Parameterized specifications

Definition 5 *A parameterized specification PSP is a pair (SP, SP') of specifications where $SP = (\Sigma, E)$ is called the* formal parameter, *$SP' = SP + (\Sigma'', E'')$ is called the* target specification, *and (Σ'', E'') is called the* body *of the parameterized specification.*

Terms built on the signature Σ and a denumerable set X_{SP} of parameter variables, *i.e. variables of sort $s \in \Sigma$, are called* parameter terms.

Example 1 *Let us axiomatize the list structure on any kind of elements. Classically "nil" is the empty list and "cons" the constructor for lists. Concatenation of lists is denoted by the function "append". In order to define a product operation on lists that computes the product of its elements, it is needed to constrain the elements to be in a monoid with an identity element. Then the formal parameter SP is the following specification* MONOID, *in which equalities have been oriented into rewrite rules:*

$$\begin{array}{rcl} sort & Elem & \\ id: & \mapsto & Elem \\ \pi : Elem, Elem & \mapsto & Elem \\ \forall e : Elem,\ \pi(e, id) & \rightarrow & e \\ \forall e : Elem,\ \pi(id, e) & \rightarrow & e \\ \forall e_1, e_2, e_3 : Elem,\ \pi(\pi(e_1, e_2), e_3) & \rightarrow & \pi(e_1, \pi(e_2, e_3)). \end{array}$$

Let us consider the following parameterized specification with the formal parameter MONOID *and the body:*

$$\begin{array}{rcl} sort & List & \\ cons : Elem, List & \mapsto & List \\ nil: & \mapsto & List \\ append : List, List & \mapsto & List \\ prod : List & \mapsto & Elem \\ \forall l : List,\ append(nil, l) & \rightarrow & l \\ \forall e : Elem, l, l' : List,\ append(cons(e, l), l') & \rightarrow & cons(e, append(l, l')) \\ \forall l : List,\ prod(nil) & \rightarrow & id \\ \forall e : Elem, l : List,\ prod(cons(e, l)) & \rightarrow & \pi(e, prod(l)). \end{array}$$

3.2 Semantics

Semantics for parameterized specifications have been widely studied, for instance in the many-sorted case in [8, 9, 24] and in the order-sorted case in [26, 12, 11]. The case of Horn clauses parameterized specifications is considered for example in [10, 21].

To give a semantics to a parameterized specification consists in associating to SP the class $ALG(SP)$ with its SP-homomorphisms, to SP' the class $ALG(SP')$ with its SP'-homomorphisms and to PSP a functor from the category $ALG(SP)$ to $ALG(SP')$.

Let $\mathcal{V}$ be the forgetful functor associated to the enrichment $SP \subseteq SP'$. From a given SP-algebra $\mathcal{A} \in ALG(SP)$, a SP'-algebra denoted $\mathcal{T}_{SP'}(\mathcal{A})$ can be explicitely built, in the following way: let

- $Const(\mathcal{A}) = \{a :\mapsto s \mid a \in \mathcal{A}_s, s \in \Sigma\}$.
- $Eqns(\mathcal{A}) = \{f(a_1, ..., a_n) = f_{\mathcal{A}}(a_1, ..., a_n) \mid \forall a_i \in \mathcal{A}, \forall f \in \Sigma\}$.
- $SP'(\mathcal{A}) = (\Sigma' \cup Const(\mathcal{A}), E' \cup Eqns(\mathcal{A}))$.
- $\mathcal{T}_{SP'}(\mathcal{A}) = \mathcal{V}_{\mathcal{A}}(\mathcal{T}_{SP'(\mathcal{A})})$, where $\mathcal{V}_{\mathcal{A}}$ is the forgetful functor from the category of the $SP'(\mathcal{A})$-algebras to the category of the SP'-algebras.

$\mathcal{T}_{SP'}(\mathcal{A})$ is *the free construction* on $\mathcal{A}$ w.r.t. $\mathcal{V}$. The concepts of free construction and free functor are precisely defined for instance in [8].

Theorem 1 *[8] Let $\mathcal{V}$ be the forgetful functor from $ALG(SP')$ to $ALG(SP)$. For any $\mathcal{A} \in ALG(SP)$, let $\mathcal{F}(\mathcal{A})$ be defined as $\mathcal{T}_{SP'}(\mathcal{A})$. Then $\mathcal{F}$ extends to a free functor from $ALG(SP)$ to $ALG(SP')$ called the* free functor *w.r.t. $\mathcal{V}$.*

Definition 6 *$\mathcal{F}$, the free functor w.r.t. $\mathcal{V}$, is the* semantics of the parameterized specification *PSP.*

Note that a first kind of genericity is obtained with the free functor: from a class $ALG(SP)$ of SP-algebras, the free functor $\mathcal{F}$ generates the class of algebras

$$\mathcal{F}(ALG(SP)) = \{\mathcal{F}(\mathcal{A}) \mid \mathcal{A} \in ALG(SP)\ \}.$$

A second kind of genericity obtained from a parameterized specification, is to generate specifications and, for this purpose, the notion of parameter passing is necessary.

3.3 Parameter passing

Parameter passing is intended to formalize the instantiation of the formal parameter specification SP into an actual specification SP_1.

Definition 7 *A* signature morphism *m from Σ to Σ_1 is a function $m : \Sigma \mapsto \Sigma_1$ such that:*

$$\forall f : s_1, ...s_n \mapsto s \in \Sigma, m(f) : m(s_1), ..., m(s_n) \mapsto m(s) \in \Sigma_1.$$

Given a signature morphism m from Σ to Σ_1 and a Σ-axiom $e = (t = t')$, the *translated axiom* denoted $m^*(e) = (m^*(t) = m^*(t'))$ is inductively defined by

$$m^*(x : s) = x : m(s) \text{ and } m^*(f(t_1, ..., t_n)) = m(f)(m^*(t_1), ..., m^*(t_n)).$$

Definition 8 *A* specification morphism *m from $SP = (\Sigma, E)$ to $SP_1 = (\Sigma_1, E_1)$ is a signature morphism from Σ to Σ_1 such that for any axiom $e \in E$, the translated axiom $m^*(e)$ is valid in the initial SP_1-algebra.*

A forgetful functor is associated to a specification morphism m.

Definition 9 *Assume that m is a specification morphism from $SP = (\Sigma, E)$ to $SP_1 = (\Sigma_1, E_1)$. Then the* forgetful functor *$\mathcal{V}_m$ from $ALG(SP_1)$ to $ALG(SP)$ is defined as follows:*

- *$\forall A' \in ALG(SP_1)$, $A = \mathcal{V}_m(A')$ is the SP-algebra such that $\forall s \in \Sigma, A_s = A'_{m(s)}$ and $\forall f \in \Sigma, f_A = m(f)_{A'}$.*
- *$\forall h'$ SP_1-morphism, $h = \mathcal{V}(h')$ is the SP-homomorphism such that $h_s = h'_{m(s)}$ for any $s \in \Sigma$.*

Given a specification morphism from the formal parameter specification to the actual parameter specification, an instantiated specification can be built.

Definition 10 *Given a parameterized specification $PSP = (SP, SP')$ and a specification morphism m from SP to SP_1, a* parameter passing *is given by:*

- *The specification morphism m' defined by*

$$\begin{aligned}
\forall s' \in \Sigma', m'(s') = \; & \text{if } s' \in S \text{ then } m(s') \text{ else } s' \\
\forall f' : s'_1, ..., s'_n \mapsto s' \in \Sigma', m'(f') = \; & \text{if } f' \in \Sigma \\
& \text{then } m(f') : m(s'_1), ..., m(s'_n) \mapsto m(s') \\
& \text{else } f' : m'(s'_1), ..., m'(s'_n) \mapsto m'(s').
\end{aligned}$$

- *The* instantiated specification $SP'_1 = SP_1 + (m'(\Sigma' - \Sigma), m'^{*}(E' - E))$.
- *The specification morphisms $p : SP \mapsto SP'$ and $p_1 : SP_1 \mapsto SP'_1$ which are inclusions.*

Notation: $\mathcal{V}_m$, $\mathcal{V}_{m'}$, $\mathcal{V}_p$ and $\mathcal{V}_{p_1}$ denote the forgetful functors [8] respectively associated to specification morphisms m, m' and inclusions p, p_1.

Paramater passing is usually represented by a parameter passing diagram.

$$\begin{tikzcd}
SP \arrow[r, "p"] \arrow[d, "m"'] & SP' \arrow[d, "m'"] & & ALG(SP) \arrow[r, shift left, "\mathcal{F}"] & ALG(SP') \arrow[l, shift left, "\mathcal{V}_p"] \arrow[d, "\mathcal{V}_{m'}", <-] \\
SP_1 \arrow[r, "p"] & SP'_1 & & ALG(SP_1) \arrow[u, "\mathcal{V}_m"] & ALG(SP'_1) \arrow[l, "\mathcal{V}_{p_1}"']
\end{tikzcd}$$

The unicity of m' and p_1 comes from the fact that a parameter passing diagram is a pushout in the category of specifications with their morphisms. This property also implies that the composition of two such diagrams is again a parameter passing diagram and this composition is associative.

Example 2 *Let us consider the parameterized specification of Example 1 and the actual parameter* NAT*:*

$$\begin{array}{rcl}
sort & Nat & \\
0 : & \mapsto & Nat \\
s : Nat & \mapsto & Nat \\
+ : Nat, Nat & \mapsto & Nat \\
\forall n : Nat,\ n + 0 & \rightarrow & n \\
\forall n, m : Nat,\ n + s(m) & \rightarrow & s(n + m).
\end{array}$$

Let m be the specification morphism defined by:

$$m(Elem) = Nat, m(id) = 0, m(\pi) = +.$$

Instantiation of MONOID *by* NAT *needs to prove that the equalities*

$$\begin{array}{rcl}
\forall n : Nat,\ n + 0 & = & n \\
\forall n : Nat,\ 0 + n & = & n \\
\forall n_1, n_2, n_3 : Nat,\ ((n_1 + n_2) + n_3 & = & n_1 + (n_2 + n_3)).
\end{array}$$

hold in the initial algebra of NAT.

The instantiated specification is:

$$\begin{array}{rcl}
sort & Nat & \\
sort & List & \\
0 : & \mapsto & Nat \\
s : Nat & \mapsto & Nat \\
+ : Nat, Nat & \mapsto & Nat \\
cons : Nat, List & \mapsto & List
\end{array}$$

$$\begin{array}{rcl}
nil : & \mapsto & List \\
append : List, List & \mapsto & List \\
prod : List & \mapsto & Nat \\
\forall n : Nat,\ n + 0 & \rightarrow & n \\
\forall n, m : Nat,\ n + s(m) & \rightarrow & s(n+m) \\
\forall l : List,\ append(nil, l) & \rightarrow & l \\
\forall n : Nat, l, l' : List,\ append(cons(n,l), l') & \rightarrow & cons(n, append(l, l')) \\
\forall l : List,\ prod(nil) & \rightarrow & 0 \\
\forall n : Nat, l : List,\ prod(cons(n,l)) & \rightarrow & n + prod(l)).
\end{array}$$

The question is now: is the syntactic construction of SP_1' compatible with the semantics respectively chosen for PSP, SP_1 and SP_1'? Here the semantics given to specifications SP_1 and SP_1' are the initial algebras of these specifications denoted respectively by $\mathcal{T}_{SP_1}$ and $\mathcal{T}_{SP_1'}$.[1] The answer is yes, provided *correctness.*

Definition 11 *The parameter passing is* correct *for a parameterized specification PSP and a specification morphism m from SP to SP_1 if*

1. $\mathcal{V}_{p_1}(\mathcal{T}_{SP_1'}) = \mathcal{T}_{SP_1}$, *property called* protection of actual parameter,
2. $\mathcal{V}_{m'}(\mathcal{T}_{SP_1'}) = \mathcal{F} \circ \mathcal{V}_m(\mathcal{T}_{SP_1})$ *property called* compatibility of parameter passing.

The first property expresses that SP_1' is a protected enrichment of SP_1. The second property expresses the fact that the semantics $\mathcal{F}$ of PSP agrees with the semantics of the instantiated specification SP_1'.

This definition of correctness is relative to one specification morphism. In order to get a notion of correctness that holds for any specification morphism, a stronger property on functors is needed.

3.4 Persistency

The correctness of parameter passing for every specification morphism requires that the functor $\mathcal{F}$ be persistent. Intuitively, persistency means that for any SP-algebra $\mathcal{A} \in ALG(SP)$, $\mathcal{A}$ is protected in $\mathcal{F}(\mathcal{A})$.

Definition 12 *[8] Given a parameterized specification PSP and the forgetful functor $\mathcal{V}_p : ALG(SP') \mapsto ALG(SP)$, the free functor $\mathcal{F} : ALG(SP) \mapsto ALG(SP')$ is said* persistent *if $\mathcal{V}_p \circ \mathcal{F} = \mathcal{I}$, where $\mathcal{I}$ is the identity functor on $ALG(SP)$, up to a natural isomorphism.*

The parameterized specification PSP is also said persistent.

Proposition 1 *[8] Given a parameterized specification PSP with a persistent functor $\mathcal{F} : ALG(SP) \mapsto ALG(SP')$ and a specification morphism m from SP to SP_1, there exists a persistent functor $\mathcal{F}_1 : ALG(SP_1) \mapsto ALG(SP_1')$, called* extension *of $\mathcal{F}$ according to m. Moreover $\mathcal{F}_1$ is uniquely defined by*

$$\mathcal{V}_{m'} \circ \mathcal{F}_1 = \mathcal{F} \circ \mathcal{V}_m \text{ and } \mathcal{V}_{p_1} \circ \mathcal{F}_1 = \mathcal{I}_1$$

where $\mathcal{I}_1$ is the identity functor on $ALG(SP_1)$.

[1]It is implicitely assumed that SP_1 and SP_1' have no empty sorts

If $\mathcal{F}$ is persistent, then for any specification morphism m, the functor $\mathcal{F}_1$ exists and is persistent. This is exactly what is needed for correctness of parameter passing, for each specification morphism.

Theorem 2 *[8] Given a parameterized specification PSP, the parameter passing is correct for PSP and any specification morphism m iff PSP is persistent in $ALG(SP)$.*

Proof: With the same notations as before:

- If PSP is persistent, $\mathcal{F}$ is persistent and $\mathcal{F}_1$ is uniquely defined by $\mathcal{V}_{p_1} \circ \mathcal{F}_1 = \mathcal{I}_1$ and $\mathcal{V}_{m'} \circ \mathcal{F}_1 = \mathcal{F} \circ \mathcal{V}_m$, according to Proposition 1. The two properties defining correctness are just obtained by applying them to the initial object of $ALG(SP_1)$, namely $\mathcal{T}_{SP_1}$.
- Conversely, assume that parameter passing is correct and let us prove that $\forall \mathcal{A} \in ALG(SP), \mathcal{V}_p \circ \mathcal{F}(\mathcal{A}) = \mathcal{A}$. Let us choose the special morphism $m_{\mathcal{A}} : SP \mapsto SP(\mathcal{A}) = SP + (\emptyset, Const(\mathcal{A}), Eqns(\mathcal{A}))$. It can be proved that $\mathcal{V}_{m_{\mathcal{A}}}(\mathcal{T}_{SP(\mathcal{A})}) = \mathcal{A}$ and thus $\mathcal{V}_p \circ \mathcal{F}(\mathcal{A}) = \mathcal{V}_p \circ \mathcal{F} \circ \mathcal{V}_{m_{\mathcal{A}}}(\mathcal{T}_{SP(\mathcal{A})}) = \mathcal{V}_p \circ \mathcal{V}_{m'_{\mathcal{A}}}(\mathcal{T}_{SP'(\mathcal{A})}) = \mathcal{V}_{m_{\mathcal{A}}} \circ \mathcal{V}_{p_1}(\mathcal{T}_{SP'(\mathcal{A})}) = \mathcal{V}_{m_{\mathcal{A}}}(\mathcal{T}_{SP(\mathcal{A})}) = \mathcal{A}$.

$\square$

Example 3 *A example of a non-persistent parameterized specification [8] is given by $PSP = (SP, SP')$ where $SP = \{\{s\}, \emptyset, \emptyset\}$ and $SP' = \{\{s\}, \{e\}, \emptyset\}$.*

Let NAT *be the usual specification of natural numbers,* NAT $= \{\{Nat\}, \{0 :\mapsto Nat, succ : Nat \mapsto Nat\}, \emptyset\}$*, consider now the actual parameter* $SP_1 =$ NAT $+\{succ(succ(x)) = x\}$*, and the specification morphism m defined by $m(s) = Nat$.*

Then the respective domains of sort Nat of $\mathcal{T}_{SP_1}$, $\mathcal{F} \circ \mathcal{V}_m(\mathcal{T}_{SP_1})$, and $\mathcal{T}_{SP'_1}$ are respectively $\{0, succ(0)\}$, $\{0, succ(0), e\}$ and $\{0, succ(0), e, succ(e)\}$. Neither the compatibility of parameter passing nor the protection of actual parameter are satisfied.

3.5 Generic algebra

We now consider different questions: how to prove correctness of parameter passing, for any specification morphism m? How to prove a generic assertion, that is, how to prove that for any specification morphism m, the translated assertion (using m) is valid in the initial algebra of the instantiated specification? Both questions have answers that need the introduction of a generic algebra.

Let $SP = (\Sigma, E)$ be a specification and $\mathcal{X}$ a denumerable set of variables whose sorts are in Σ. Let $\mathcal{T}_{SP}(\mathcal{X})$ be the initial term algebra associated to the specification $(\Sigma \cup \mathcal{X}, E)$. $\mathcal{T}_{SP}(\mathcal{X})$ is a Σ-algebra whose carrier is the quotient $\mathcal{T}(\Sigma \cup \mathcal{X})/E$ of the set of terms $\mathcal{T}(\Sigma \cup \mathcal{X})$ by the congruence $\xleftrightarrow{*}_E$. Note that if $\mathcal{X}$ is any set of variables with sorts in Σ, $\mathcal{T}_{SP}(\mathcal{X})$ is the free SP-algebra generated by $\mathcal{X}$. $\mathcal{T}_{SP}(\emptyset)$ is the initial SP-algebra. The *PSP-generic algebra* is obtained for a third choice of $\mathcal{X}$:[2]

Definition 13 *Let $PSP = (SP, SP')$ be a parameterized specification and X_{SP} a set of variables of parameter sorts. The* PSP-generic algebra *is the Σ'-algebra $\mathcal{T}_{SP'}(X_{SP})$, whose carrier $\mathcal{T}(\Sigma' \cup X_{SP})/E'$ is the quotient by $\xleftrightarrow{*}_{E'}$ of the set $\mathcal{T}(\Sigma' \cup X_{SP})$ of* PSP-generic terms.

[2]The word *generic* is currently used for free algebras [29] but less often in the context of parameterization [7].

3.6 Generic theory of a parameterized specification

The set of theorems valid in the class of algebras $\mathcal{F}(ALG(SP))$ associated to a parameterized specification, defines the *generic theory* of the parameterized specification.

Definition 14 *The* generic theory *of the parameterized specification PSP denoted $Th(PSP)$ is the set*

$$\{(\forall X, t = t') | \forall A \in ALG(SP), \mathcal{F}(A) \models (\forall X, t = t')\}.$$

The generic theory is also called equational theory of PSP in [24]. Note that in the degenerated case where SP is the empty specification, then $Th(PSP)$ is the inductive theory of SP'.

Definition 15 *Any substitution $\sigma : X \mapsto \mathcal{T}(\Sigma' \cup X_{SP})$, is called a* PSP-generic substitution.

As a consequence of previous definitions, an equality $(\forall X, t = t')$ holds in the PSP-generic algebra $\mathcal{T}_{SP'}(X_{SP})$, which is denoted by $\mathcal{T}_{SP'}(X_{SP}) \models (\forall X, t = t')$, if for any PSP-generic substitution $\sigma : X \mapsto \mathcal{T}(\Sigma' \cup X_{SP})$, $\sigma(t) \overset{*}{\longleftrightarrow}_E \sigma(t')$.

Example 4 *Let us consider again the parameterized specification of Example 1.*

The term append(cons(e, nil), nil) with e a variable of sort Elem, is a PSP-generic term.

Substitutions $(l \mapsto nil)$, $(l \mapsto cons(e, nil))$, $(l \mapsto append(cons(e, nil), nil))$ are PSP-generic substitutions.

The equality $(\forall e : Elem, l : List,\ append(cons(e, nil), l) = cons(e, l))$ holds in the PSP-generic algebra, just because it holds in the whole class of SP'-algebras.

The next theorem states that the generic theory is exactly the set of theorems valid in the PSP-generic algebra.

Theorem 3 *[24] Let $PSP = (SP, SP')$ be a parameterized specification, X_{SP} a set of variables of parameter sorts and $\mathcal{T}_{SP'}(X_{SP})$ the PSP-generic algebra.*

$$\mathcal{T}_{SP'}(X_{SP}) \models (\forall X, t = t') \text{ iff } (\forall X, t = t') \in Th(PSP).$$

Proof: (Sketch) If $(\forall X, t = t') \in Th(PSP)$, then it holds in $\mathcal{F}(\mathcal{T}_{SP}(X_{SP}))$ that is isomorphic to $\mathcal{T}_{SP'}(X_{SP})$.

Conversely, assume that $\mathcal{T}_{SP'}(X_{SP}) \models (\forall X, t = t')$. To prove that the equality holds in $\mathcal{F}(\mathcal{A})$ for any $\mathcal{A}$, let us prove that it holds in $\mathcal{T}_{SP'}(\mathcal{A})$ which is isomorphic to $\mathcal{F}(\mathcal{A})$. Any assignment $\sigma : X \mapsto \mathcal{T}_{SP'}(\mathcal{A})$ can be decomposed into $\mu : X \mapsto \mathcal{T}_{SP'}(X_{SP})$ and $\alpha : \mathcal{T}_{SP'}(X_{SP}) \mapsto \mathcal{T}_{SP'}(\mathcal{A})$. Then from $\mu(t) = \mu(t')$, it is easily deduced that $\sigma(t) = \alpha(\mu(t)) = \alpha(\mu(t')) = \sigma(t')$. $\square$

The following result explains in which sense a theorem in $Th(PSP)$ is generic: validity of an equality in the PSP-generic algebra means validity of the translated equality in any instantiation of the parameterized specification, provided that the parameterized specification is persistent. A similar result is given in [7].

Theorem 4 *Let $PSP = (SP, SP')$ be a persistent parameterized specification and X_{SP} a set of variables of parameter sorts. Then the two following properties are equivalent:*

1. $\mathcal{T}_{SP'}(X_{SP}) \models (\forall X, t = t')$

2. *for any specification morphism m, $\mathcal{T}_{SP'_1} \models (\forall m'^*(X), m'^*(t) = m'^*(t'))$.*

Proof: If $\mathcal{T}_{SP'}(X_{SP}) \models (\forall X, t = t')$, then the equality holds in $\mathcal{F}(\mathcal{A})$ for any $\mathcal{A}$, for instance in $\mathcal{F} \circ \mathcal{V}_m(\mathcal{T}_{SP_1})$ for any specification morphism m. Assuming persistency, the compatibility of parameter passing for m yields $\mathcal{F} \circ \mathcal{V}_m(\mathcal{T}_{SP_1}) = \mathcal{V}_{m'}(\mathcal{T}_{SP'_1})$. Then $\mathcal{V}_{m'}(\mathcal{T}_{SP'_1}) \models (\forall X, t = t')$, so $\mathcal{T}_{SP'_1} \models (\forall m'^*(X), m'^*(t) = m'^*(t'))$.

Conversely, for any $\mathcal{A} \in ALG(SP)$, there exists a specification morphism $m_{\mathcal{A}}$ and a specification $SP(\mathcal{A})$ such that $\mathcal{A} = \mathcal{V}_{m_{\mathcal{A}}}(\mathcal{T}_{SP(\mathcal{A})})$. Since $\mathcal{T}_{SP'(\mathcal{A})} \models (\forall m'^*_{\mathcal{A}}(X), m'^*_{\mathcal{A}}(t) = m'^*_{\mathcal{A}}(t'))$ implies $\mathcal{V}_{m'_{\mathcal{A}}}(\mathcal{T}_{(SP'(\mathcal{A}))}) \models (\forall X, t = t')$, then $\mathcal{F} \circ \mathcal{V}_{m_{\mathcal{A}}}(\mathcal{T}_{SP(\mathcal{A})}) \models (\forall X, t = t')$ and $\mathcal{F}(\mathcal{A}) \models (\forall X, t = t')$. By Theorem 3, $\mathcal{T}_{SP'}(X_{SP}) \models (\forall X, t = t')$. $\square$

Actually only the compatibility of parameter passing is used in this proof. But this property is equivalent to persistency, as shown in [23].

The next theorem relates the notion of persistency with proof theoretical properties.

Theorem 5 *[9] Let $PSP = (SP, SP')$ be a parameterized specification and X_{SP} a set of variables of parameter sorts. PSP is persistent iff the following two properties are satisfied:*

1. *$PSP = (SP, SP')$ is* generic sufficiently complete, *i.e.: $\forall t \in \mathcal{T}(\Sigma' \cup X_{SP})$, t of parameter sort, $\exists t_0 \in \mathcal{T}(\Sigma \cup X_{SP})$ such that $t \xleftrightarrow{*}_{E'} t_0$.*

2. *$PSP = (SP, SP')$ is* generic consistent, *i.e.: $\forall t, t' \in \mathcal{T}(\Sigma \cup X_{SP})$ of parameter sorts, $t \xleftrightarrow{*}_{E'} t'$ iff $t \xleftrightarrow{*}_{E} t'$.*

This definition expresses in other words that the enrichment $(\Sigma \cup X_{SP}, E) \subseteq (\Sigma' \cup X_{SP}, E')$ is protected.

In order to go further and design effective tools for parameterized proofs, we now focus on equational theories described by rewrite systems.

4 Generic ground reducibility

In order to check persistency of a parameterized specification and validity in the PSP-generic algebra, the notion of PSP-generic ground reducibility is needed.

Let $PSP = (SP, SP')$ be a parameterized specification with $SP = (\Sigma, R)$ and $SP' = (\Sigma', R')$ where R and R' are rewrite systems. Let X_{SP} be an infinite set of variables of parameter sorts.

Definition 16 *Given a rewrite system R', terminating on $\mathcal{T}(\Sigma' \cup X_{SP})$, a term $t \in \mathcal{T}(\Sigma' \cup X)$ is* PSP-generic ground reducible *with R' if for any PSP-generic substitution $\sigma : X \mapsto \mathcal{T}(\Sigma' \cup X_{SP})$, $\sigma(t)$ is reducible using R'.*

An equality $(\forall X, t = t')$ is PSP-generic ground reducible *with R' if for any PSP-generic substitution $\sigma : X \mapsto \mathcal{T}(\Sigma' \cup X_{SP})$, such that $\sigma(t) \neq \sigma(t')$, either $\sigma(t)$ or $\sigma(t')$ is reducible using R'.*

Algorithms for checking ground reducibility can be extended to check PSP-generic ground reducibility. Here, the test for ground reducibility in the case of left-linear rules given in [15] is generalized to a test for generic ground reducibility. The goal is to exhibit a finite set of substitutions $\mathcal{S}$ such that a term is generic ground reducible iff all its instances by substitutions in $\mathcal{S}$ are reducible. In the case of left-linear rules in R', the idea to construct $\mathcal{S}$ is that left-hand sides of rules have a finite depth which bounds the number of substitutions to be tested. Some preliminary definitions are needed [15].

The length of a term t is the maximal size of positions ω in $\mathcal{D}(t)$. Let $d = depth(R')$ be the maximal depth of left-hand sides of rules in R'. The top of a term t at depth i is a term defined by:
$top(t, i) = t$ if $depth(t) < i$
$top(f(t_1, ..., t_n), 0) = f(x_1, ..., x_n)$ where x_i are new variables
$top(f(t_1, ..., t_n), i) = f(top(t_1, i-1), ..., top(t_n, i-1))$
for any symbol $f \in \mathcal{F}$.

Given a set of rewrite rules R' of depth d, let

$$\mathcal{S}(R') = \{\ top(t_0, d) \mid t_0 \text{ is an } R'\text{-irreducible PSP-generic term }\}$$

be called the *PSP-generic test set.* For practical reasons, variables in terms of $\mathcal{S}(R')$ are assumed distinct, which is always possible by renaming them: $\forall t, t' \in \mathcal{S}(R')$, $\mathcal{V}(t) \cap \mathcal{V}(t') = \emptyset$. The set $\mathcal{S}(R')$ is computed as the limit of a stationary sequence of sets $\mathcal{S}_i$ defined as follows:

$$\mathcal{S}_i = \{top(t_0, d) \mid t_0 \text{ is an } R'\text{-irreducible PSP-generic term such that } depth(t_0) \leq i\}.$$

Then $\mathcal{S}(R') = \mathcal{S}_k$ as soon as $\mathcal{S}_k = \mathcal{S}_{k+1}$ for some k.

Theorem 6 *A term t is PSP-generic ground reducible by a left-linear rewrite system R' iff all its instances*

$$\{\sigma(t) \mid \sigma : \mathcal{V}(t) \mapsto \mathcal{S}(R')\},$$

obtained by substituting variables of t by terms in $\mathcal{S}(R')$, are reducible by R'.

Proof: The proof is similar to the proof in [16].

- Assume that all instances of t obtained by substituting variables of t by terms in $\mathcal{S}(R')$, are reducible by R'. For any PSP-generic substitution σ', if σ' is not R'-normalized, then $\sigma'(t)$ is R'-reducible. Otherwise, let us define for any variable x of t, $\sigma(x) = top(\sigma'(x), d)$. Then $\sigma \in \mathcal{S}(R')$ and $\sigma(t)$ is R'-reducible by hypothesis. Since σ' is an instance of σ, $\sigma'(t)$ is also R'-reducible.
- Assume now that t is PSP-generic ground reducible. Given $\sigma \in \mathcal{S}(R')$, let us define for any variable x_i of t, $t_i = \sigma(x_i)$. Then there exists t'_i an R'-irreducible instance of t_i such that $t_i = top(t'_i, d)$. Finally let us define σ' such that $\sigma'(x_i) = t'_i$. Since t is PSP-generic ground reducible, $\sigma'(t)$ is R'-reducible by a rewrite rule $l \rightarrow r$, at some non-variable position ω in t because σ' is normalized. But for any position v in l, the top symbols of $l_{|v}$, $\sigma(t)_{|\omega.v}$ and $\sigma'(t)_{|\omega.v}$ are the same, because of the definition of d and the fact that $t_i = top(t'_i, d)$. Now let W be the set of variable occurrences in l and for any variable y at occurrence $v \in W$ define $\sigma''(y) = \sigma(t)_{|\omega.v}$. Note that σ'' is well-defined because y has only one occurrence in l. So $\sigma(t)_{|\omega} = \sigma''(l)$ and so $\sigma(t)$ is R'-reducible.

□

Example 5 *Consider the parameterized specification of Example 1. In order to check the PSP-generic ground reducibility of $prod(append(l, l'))$, the following generic test set needs to be considered:*

$$\{nil, cons(e_0, nil), cons(e_1, cons(e_2, nil))\}.$$

where e_0, e_1, e_2 are new variables of sort Elem in X_{SP}. For each deduced substitution α, $\alpha(t)$ is reducible using R'.

5 Proof of persistency

5.1 Proof of generic sufficient completeness

We now extend the proof of sufficient completeness for convergent rewrite systems of Kapur, Narendran and Zhang in [17]. Their result states the equivalence between sufficient completeness and a check for ground reducibility, provided that terms built on the imported signature are preserved.

Definition 17 *Let $PSP = (SP, SP')$ be a parameterized specification such that $SP = (\Sigma, R)$ and $SP' = (\Sigma', R')$. R'* preserves parameters *if $\forall(l \rightarrow r) \in R'$, whenever l has a parameter sort, r has a parameter sort, and whenever l is a parameter term, r is a parameter term.*

Proposition 2 *Let $PSP = (SP, SP')$ be a parameterized specification such that $SP = (\Sigma, R) \subseteq SP' = (\Sigma', R')$, X_{SP} is a denumerable set of variables of parameter sorts, and R' is a convergent rewrite system on $\mathcal{T}(\Sigma' \cup X_{SP})$ preserving parameters.*
Then the following propositions are equivalent:

1. *$\forall f \in \Sigma' - \Sigma$, whose range is a sort $s \in \Sigma$, $f(x_1, ..., x_n)$ is PSP-generic ground reducible with R',*
2. *$\forall t' \in \mathcal{T}(\Sigma' \cup X_{SP})$ of sort $s \in \Sigma$, $\exists t \in \mathcal{T}(\Sigma \cup X_{SP})$ such that $t \overset{*}{\longleftrightarrow}_{R'} t'$.*

Proof: The proof is an extension of [17].

- Let us first prove that (1) implies (2).
Consider a term $t' \in \mathcal{T}(\Sigma' \cup X_{SP})$ of sort $s \in \Sigma$ and its R'-irreducible form t''. Either $t'' \in \mathcal{T}(\Sigma \cup X_{SP})$ and then $t = t''$, or $t'' \notin \mathcal{T}(\Sigma \cup X_{SP})$. In this case, t'' contains a subterm $f(u_1, ..., u_n)$, with $u_1, ..., u_n \in \mathcal{T}(\Sigma' \cup X_{SP})$, $f \in \Sigma' - \Sigma$ with a co-arity $s' \in \Sigma$. (Otherwise, all subterms of t'' would be of sort $s'' \in \Sigma' - \Sigma$, including t'' itself. This is impossible because the term t of sort $s \in \Sigma$ cannot be rewritten to t'' of sort $s'' \in \Sigma' - \Sigma$, if $\forall(l \rightarrow r) \in R'$, whenever l has a parameter sort, r has a parameter sort.) The subterm $u = f(u_1, ..., u_n)$ is indeed a PSP-generic instance of $f(x_1, .., x_n)$, so is reducible for R', which contradicts the hypothesis that t'' is R'-irreducible.
- Let us now prove that (2) implies (1).
If $f(x_1, .., x_n)$ is not PSP-generic ground reducible for R', there exists a PSP-generic substitution σ such that $t' = f(\sigma(x_1), .., \sigma(x_n))$ is irreducible for R'. Assume now that $\exists t \in \mathcal{T}(\Sigma \cup X_{SP})$ of sort $s \in \Sigma$, such that $t \overset{*}{\longleftrightarrow}_{R'} t'$. Since R' is convergent on $\mathcal{T}(\Sigma \cup X_{SP})$, this implies that $t \overset{*}{\longrightarrow}_{R'} t'$. Which is impossible if R' preserves parameters.

□

Note that the property of preserving parameters is very simple to achieve in most cases. It is satisfied for instance when every rule in $R' - R$ contains in its left-hand side at least a function symbol of $\Sigma' - \Sigma$. In a structured programming methodology and especially in parameterized specifications, this hypothesis does not appear as a restriction.

5.2 Proof of generic consistency

In the case where the basic specification is given with a ground convergent rewrite systems, the completion process appears as an interesting tool to both prove consistency of an enrichment and produce simultaneously a ground convergent rewrite system for the enriched specification. Given an enrichment $SP = (\Sigma, R) \subseteq SP' = (\Sigma', E')$ with R a ground convergent rewrite system on $\mathcal{T}(\Sigma)$. The general idea is to complete E' into a ground convergent system R' on $\mathcal{T}(\Sigma')$ and to check that whenever a rewrite rule, whose left and right-hand sides both belong to $\mathcal{T}(\Sigma)$, is added, then this rule is an inductive consequence of R.

We design a similar consistency proof procedure in the slightly more general framework of a parameterized specification PSP with $SP = (\Sigma, R) \subseteq SP' = (\Sigma', E')$ with R a convergent rewrite system on $\mathcal{T}(\Sigma \cup X_{SP})$. (This can be checked by completion). PSP is generic consistent if whenever an equality, whose left and right-hand sides both belong to $\mathcal{T}(\Sigma \cup X_{SP})$, is added, then this is a theorem valid in $ALG(SP)$, that is $t \overset{*}{\longleftrightarrow}_R t'$. In order to detect inconsistencies, we need the following definition:

Definition 18 *Let us consider a parameterized specification PSP with* $SP = (\Sigma, R) \subseteq SP' = (\Sigma', E')$ *with R a convergent rewrite system. A set of equalities C is* provably inconsistent with *$Th(SP)$ if it contains an equality $(\forall X, t = t')$ such that t and t' are in $\mathcal{T}(\Sigma \cup X_{SP})$ and are not $\overset{*}{\longleftrightarrow}_R$-equivalent.*

If SP is itself a parameterized specification, say $SP = (SP_0, SP_1)$, then $Th(SP)$ is its generic theory and the previous definition must be refined by replacing $\overset{*}{\longleftrightarrow}_R$-equivalence by validity in the SP_0-generic algebra.

In the completion process described below, it is convenient to split the set of equalities $E' - R$ into two parts: one, called C, which contains only parameter equalities (i.e. built on terms of $\mathcal{T}(\Sigma \cup X_{SP})$), and the other, called P, that contains non-parameter ones. $OCP(P \cup R, P)$ denote the set of ordered critical pairs [1, 3] obtained by superposition of $P \cup R$ on P. There is no need to superpose rules in R with themselves. There is also no need to try superpositions of P on R because by construction, terms involved in P contain at least one function symbol in $\Sigma' - \Sigma$. $CP(R, C)$ as usual denote the set of critical pairs obtained by superposition of R on C and conversely. Note that whenever an equality contains only parameter terms and can be superposed with R, then the equality is dropped in C and such superpositions are taken into account in $CP(R, C)$.

Let P be a set of equalities (quantified pairs of terms), C a set of conjectures (parameter equalities), R the underlying rewrite system on parameter terms, terminating and confluent on $\mathcal{T}(\Sigma \cup X_{SP})$, and $>$ a reduction ordering that contains R and can be extended to a reduction ordering on $\mathcal{T}(\Sigma' \cup X_{SP})$ total on E'-equivalence classes. The generic consistency proof procedure is expressed by the following set $\mathcal{GC}$ of inference rules, in which R is implicit:

1. Deduce
$$\frac{P, C}{P \cup \{p = q\}, C} \quad \text{if } (p, q) \in OCP(P \cup R, P)$$

2. Deflation
$$\frac{P \cup \{p = q\}, C}{P, C \cup \{p = q\}} \quad \text{if } p \textit{ and } q \textit{ parameter terms}$$

3. Delete
$$\frac{P \cup \{p = p\}, C}{P, C}$$

4. Collapse
$$\frac{P \cup \{p = q\}, C}{P \cup \{p' = q\}, C} \quad \text{if } (p \xrightarrow{l \to r}_{R \cup P>} p' \ \&\ p \sqsupset l) \text{ or } (p \to_{R \cup P>} p' \ \&\ q > p')$$

5. Deduce conjecture
$$\frac{P, C}{P, C \cup \{p = q\}} \quad \text{if } (p, q) \in CP(R, C)$$

6. Delete conjecture
$$\frac{P, C \cup \{p = q\}}{P, C} \quad \text{if } (p = q) \in Th(SP)$$

7. Simplify
$$\frac{P, C \cup \{p = q\}}{P, C \cup \{p' = q\}} \quad \text{if } p > p' \ \&\ p \xleftrightarrow{+}_R p'$$

8. Compose
$$\frac{P, C \cup \{p = q\}}{P, C \cup \{p' = q\}} \quad \text{if } p > p' \ \&\ p \longleftrightarrow_C^{g=d} p' \ \&\ p \sqsupset g > d$$

9. Generic consistency Disproof
$$\frac{P, C \cup (p = q)}{Disproof} \quad \text{if } (p = q) \ \textit{provably inconsistent with } Th(SP)$$

In these inference rules, $\sqsupset$ denotes the strict encompassment ordering defined by $t \sqsupset t'$ if $t_{|\omega} = \sigma(t')$, for some position ω of t and some substitution σ, with $\omega \neq \epsilon$ or $\sigma \neq Id$.

Note that this set of inference rules is more general than the one for unfailing completion [1], obtained as a subset when $C = \emptyset$, and the one for proof by consistency [1], obtained for $P = \emptyset$. The *Deflation* inference rule is used first to split the set of equalities $E' - E$ added in the enrichment, into P and C such that C contains only parameter equalities and P_0 contains all other equalities.

Lemma 1 *If $(P, C) \vdash (P', C')$, then the congruences $\xleftrightarrow{*}_{P \cup C \cup R}$ and $\xleftrightarrow{*}_{P' \cup C' \cup R}$ coincide on $\mathcal{T}(\Sigma' \cup X_{SP})$.*

Proof: by looking at the different inference rules. □

The considered set of proofs are proofs on PSP-generic terms using equalities in $P \cup C \cup R$. The goal is to transform such a proof $t \xleftrightarrow{*}_{P \cup C \cup R} t'$ into a proof of the form

$$t \xrightarrow{*}_{R \cup P>} t'' \xleftarrow{*}_{R \cup P>} t'$$

or to find a provable inconsistency with $Th(SP)$.

The proof reduction relation that reflects the inference rule system $\mathcal{GC}$, is now defined.

1. <u>Deduce:</u> $t' \leftarrow_{P> \cup R}^{g=d} t \rightarrow_{P>}^{l=r} t'' \Longrightarrow t' \longleftrightarrow_P^{p=q} t''$

2. <u>Deflation:</u> $t \longleftrightarrow_P^{p=q} t' \Longrightarrow t \longleftrightarrow_C^{p=q} t'$

3. <u>Delete:</u> $t \longleftrightarrow_P^{p=p} t' \Longrightarrow \Lambda$ where Λ is the empty proof.

4. <u>Collapse:</u> $t \longleftrightarrow_P^{p=q} t' \Longrightarrow t \rightarrow_{P> \cup R}^{l \rightarrow r} t'' \longleftrightarrow_P^{p'=q} t'$.

5. <u>Deduce conjecture:</u> $t' \leftarrow_R^{l \rightarrow r} t \longleftrightarrow_C^{g=d} t'' \Longrightarrow t' \longleftrightarrow_C^{p=q} t''$

6. <u>Delete conjecture:</u> $t \longleftrightarrow_C^{p=q} t' \Longrightarrow t \xleftrightarrow{*}_R t'$

7. Simplify: $t \longleftrightarrow_{C}^{p=q} t' \Longrightarrow t \overset{+}{\longleftrightarrow}_{R} t'' \longleftrightarrow_{C}^{p'=q} t'$

8. Compose: $t \longleftrightarrow_{C}^{p=q} t' \Longrightarrow t \longleftrightarrow_{C}^{g=d} t'' \longleftrightarrow_{C}^{p'=q} t'$

9. Peak without overlap: $t' \leftarrow_{R\cup P>}^{g=d} t \rightarrow_{P>}^{l=r} t'' \Longrightarrow t' \rightarrow_{P>}^{l=r} t_1 \leftarrow_{R\cup P>}^{g=d} t''$

10. Peak with variable overlap: $t' \leftarrow_{R\cup P>}^{g=d} t \rightarrow_{P>}^{l=r} t'' \Longrightarrow t' \overset{*}{\longrightarrow}_{P>} t_1 \overset{*}{\longleftarrow}_{R\cup P>} t''$

Lemma 2 *If $>$ is a reduction ordering that can be extended to a reduction ordering $>$ on $\mathcal{T}(\Sigma' \cup X_{SP})$ total on E'-equivalence classes, then the proof reduction relation is noetherian.*

Proof: Let us define the complexity measure of elementary proof steps by:

$$\begin{array}{rcll}
c(s \longleftrightarrow_{P}^{g=d} t) & = & (s,g,t,2) & if \quad s > t \\
c(s \longleftrightarrow_{P}^{g=d} t) & = & (t,d,s,2) & if \quad t > s \\
c(s \longleftrightarrow_{C}^{g=d} t) & = & (s,g,t,1) & if \quad s > t \\
c(s \longleftrightarrow_{C}^{g=d} t) & = & (t,d,s,1) & if \quad t > s \\
c(s \rightarrow_{R}^{g\rightarrow d} t) & = & (s,g,t,0) &
\end{array}$$

where $s, t \in \mathcal{T}(\Sigma' \cup X_{SP})$. Let $> \cup \rhd$ denote the union of the reduction ordering $>$ and the strict subterm ordering $\rhd$. Complexities of elementary proof steps are compared using the lexicographic combination, denoted $>_{ec}$ of $> \cup \rhd$, $\sqsupset$, $> \cup \rhd$ and the standard ordering on natural numbers. The complexity of a non-elementary proof is the multiset of the complexities of elementary proof steps that it contains. Complexities of non-elementary proofs are compared using the multiset extension $>_c$ of $>_{ec}$. For any proof reduction rule $L \Longrightarrow R$ defining the proof reduction relation, $c(L) >_c c(R)$. □

Theorem 7 *Let us consider a parameterized specification PSP with $SP = (\Sigma, E) \subsetneq SP' = (\Sigma', E')$ with R a convergent rewrite system on $\mathcal{T}(\Sigma \cup X_{SP})$ presenting E.*
Let $>$ be a reduction ordering that contains R and can be extended to a reduction ordering $>$ on $\mathcal{T}(\Sigma' \cup X_{SP})$, total on E'-equivalence classes, and such that no parameter term is greater than a non-parameter one.
Let $(P_0, C_0) = (E', \emptyset) \vdash (P_1, C_1) \vdash \ldots.$ be a derivation using $\mathcal{GC}$, $P_* = \bigcup_{i\geq 0} P_i$, $C_* = \bigcup_{i\geq 0} C_i$, $P_\infty = \bigcup_{i\geq 0} \bigcap_{j>i} P_j$, $C_\infty = \bigcup_{i\geq 0} \bigcap_{j>i} C_j$. Assume that $OCP(P_\infty \cup R, P_\infty) \cup CP(R, C_\infty)$ is a subset of $P_* \cup C_*$.

- *If C_∞ is empty, $(P_\infty \cup R)$ is Church-Rosser with respect to $>$ on $\mathcal{T}(\Sigma' \cup X_{SP})$ and the parameterized specification is generic consistent.*

- *If a provable inconsistency with $Th(SP)$ has been detected, the parameterized specification is not generic consistent.*

Proof: • Assume that C_∞ is empty. Let us prove by induction on $\Longrightarrow$ that for any $i \geq 0$, for any proof $t \overset{*}{\longleftrightarrow}_{P_i \cup C_i \cup R} t'$ where $t, t' \in \mathcal{T}(\Sigma' \cup X_{SP})$, there exists a proof

$$t \overset{*}{\longrightarrow}_{R\cup P_\infty^{\geq}} t'' \overset{*}{\longleftarrow}_{R\cup P_\infty^{\geq}} t'.$$

Since C_∞ is empty, this means that any C_i-equality step has been replaced by R-equality steps. So if the proof $t \overset{*}{\longleftrightarrow}_{P_i \cup C_i \cup R} t'$ is not already of the desired form, then either it contains a peak of $R \cup P_\infty$, or a non-persisting equality in P. In both cases, the proof is reducible by $\Longrightarrow$ into $t \overset{*}{\longleftrightarrow}_{P_j \cup C_j \cup R} t'$ and the induction hypothesis gives the result.

Given two terms, t and t' of $\mathcal{T}(\Sigma \cup X_{SP})$ of parameter sorts, such that $t \overset{*}{\longleftrightarrow}_{P_0 \cup C_0 \cup R} t'$, there exists a proof

$$t \overset{*}{\longrightarrow}_{R \cup P^{\gtrsim}_{\infty}} t'' \overset{*}{\longleftarrow}_{R \cup P^{\gtrsim}_{\infty}} t'.$$

But since t and t' are parameter terms and greater then any other term occurring in the proof, all these terms must be parameter terms. So no equality in P_{∞} can apply since P_{∞} contains non-parameter terms. So we get $t \overset{*}{\longleftrightarrow}_R t'$.

- If a provable inconsistency with $Th(SP)$ has been detected, this means that there exist $g, d \in \mathcal{T}(\Sigma \cup X_{SP})$ such that $g \longleftrightarrow_{C_k} d$, for some k, but do not satisfy $g \overset{*}{\longleftrightarrow}_R d$. So g and d are $\overset{*}{\longleftrightarrow}_{P_0 \cup C_0 \cup R}$-equivalent but not $\overset{*}{\longleftrightarrow}_R$-equivalent, which proves that the parameterized specification is not generic consistent.

□

Example 6 *The parameterized specification of Example 1 is persistent.*

The first step is to prove that the enrichment $(\Sigma \cup X_{SP}, R) \subseteq (\Sigma' \cup X_{SP}, R')$ is generic consistent. For that, the consistency proof procedure is applied. Since there is no critical pair, the enrichment is obviously consistent.

Second, in order to prove that the enrichment $(\Sigma \cup X_{SP}, R) \subseteq (\Sigma' \cup X_{SP}, R')$ is generic sufficiently complete, we check that R' preserves parameters.

Then we have to check that prod(l) is PSP-generic ground reducible with R'. The instantiations to be checked are:

$$\begin{array}{lll} (l & \mapsto & nil) \\ (l & \mapsto & cons(e_0, nil)) \\ (l & \mapsto & cons(e_1, cons(e_2, l))) \end{array}$$

where $e_0, e_1, e_2 \in X_{SP}$ and l is a variable of sort List. For these three instantiations σ, the term $\sigma(prod(l))$ is clearly reducible.

6 Proof of an equational theorem in the theory of a parameterized specification

In the context of proving and disproving theorems in a parameterized specification, we need a slightly different notion of provable inconsistency, that is directly inspired by the one used in [2]. We assume in this section that PSP is a parameterized specification defined by the formal parameter $SP = (\Sigma, R)$ and $SP' = (\Sigma', R')$, where R and R' are rewrite systems.

Definition 19 *Let us consider a parameterized specification PSP with $SP = (\Sigma, R) \subseteq SP' = (\Sigma', R')$, where R' is a terminating rewrite system and $>$ a reduction ordering that contains R'. A set of equalities C is* provably inconsistent with $Th(PSP)$ *if it contains an equality $(\forall X, t = t')$ which satisfies either $t > t'$ and t is not PSP-generic ground reducible, or $(\forall X, t = t')$ is not PSP-generic ground reducible.*

Replacing the notion of provable inconsistency by the notion of provable inconsistency with $Th(PSP)$ allows applying the proof by consistency method to the proof of theorems in $Th(PSP)$.

Let C be a set of conjectures, R' the underlying rewrite system of the parameterized specification, terminating and confluent on $\mathcal{T}(\Sigma' \cup X_{SP})$, and $>$ a reduction ordering that contains R'. The generic proof procedure is expressed by the following set $\mathcal{GI}$ of inference rules, that is a subset of $\mathcal{GC}$, obtained with $P = \emptyset$.

1. Deduce conjecture
$\dfrac{C}{C \cup \{p = q\}}$ if $(p, q) \in CP(R', C)$

2. Delete conjecture
$\dfrac{C \cup \{p = q\}}{C}$ if $(p = q) \in Th(PSP)$

3. Simplify
$\dfrac{C \cup \{p = q\}}{C \cup \{p' = q\}}$ if $p > p'$ & $p \xleftrightarrow{+}_{R'} p'$

4. Compose
$\dfrac{C \cup \{p = q\}}{C \cup \{p' = q\}}$ if $p > p'$ & $p \longleftrightarrow_{C}^{g=d} p'$ & $p \sqsupset g > d$

5. Generic consistency disproof
$\dfrac{C \cup (p = q)}{Disproof}$ if $(p = q)$ *provably inconsistent with* $Th(PSP)$

Theorem 8 *Let $PSP = (SP, SP')$ be a parameterized specification and X_{SP} a set of variables of parameter sorts. Let R' be a terminating and confluent rewrite system on $\mathcal{T}(\Sigma' \cup X_{SP})$ presenting E', C_0 be the set of PSP-generic conjectures to be proved, and $>$ be a reduction ordering that contains R'. Let $C_0 \vdash C_1 \vdash \ldots$. be a derivation using $\mathcal{GI}$ such that $CP(R', C_\infty)$ is a subset of C_*. If no provable inconsistency with $Th(PSP)$ has been detected, then C_0 is included in $Th(PSP)$.*

Proof: It is a consequence of the proof by consistency method in $\mathcal{T}_{SP'}(X_{SP})$ [1, 2]. □

Example 7 *In the parameterized specification of Example 1, let us prove that the PSP-generic conjecture*

$$\forall l, l' : List,\ prod(append(l, l')) = \pi(prod(l), prod(l'))$$

is valid.

This conjecture is PSP-generic ground reducible. By choosing a precedence such that $prod > \pi$, the left-hand side is bigger than the right-hand side. Superpositions on the bigger term are enough. So two superpositions have to be computed:
- With the rule $\forall l : List,\ append(nil, l) \rightarrow l$, we get

$$prod(l') = \pi(prod(nil), prod(l')).$$

The term $\pi(prod(nil), prod(l'))$ reduces to $prod(l')$ and the conjecture becomes a trivial equality.
- With the rule $\forall e : Elem, l, l' : List,\ append(cons(e, l), l') \rightarrow cons(e, append(l, l'))$, we get

$$prod(cons(e, append(l_1, l_2))) = \pi(prod(cons(e, l_1)), prod(l_2)).$$

After reduction on both sides, we get

$$\pi(e, prod(append(l_1, l_2))) = \pi(e, \pi(prod(l_1), prod(l_2)))$$

that can be simplified using the initial conjecture.

7 Conclusion

Proofs in parameterized specifications have been experimented in several systems like CEC [10] or Reveur4 [28]. To some extent, this paper may be understood as a clarification of the concepts underlying these provers.

Now several research directions can be outlined for improving this work.

- Persistency is a very strong property that sometimes one may want to drop. So a first question that arises is the following: which results remain true if persistency is not assumed? Actually persistency is assumed in Theorem 4, and only the compatibility of parameter passing is used in its proof. Theorem 4 could be weakened as follows: if $\mathcal{T}_{SP'}(X_{SP}) \models (\forall X, t = t')$, then for any specification morphism m satisfying the compatibility of parameter passing, $\mathcal{T}_{SP'_1} \models (\forall m'^*(X), m'^*(t) = m'^*(t'))$. However the problem of checking the compatibility of parameter passing, even for a specific specification morphism, is not solved. Moreover assuming compatibility of parameter passing for every specification morphism has been proved equivalent to persistency in [23].

- This work focussed on equational parameterized specifications and has to be extended to conditional specifications: mainly, this can be done along the lines of Navarro and Orejas [21], but an adequate notion of persistency needs the introduction of so-called LOG-algebras, that have an initial boolean domain, together with a property of persistency with respect to booleans. Note that Theorem 5 has been proved by Ganzinger [9] in the case where the considered class of algebras is the class $ALG(SP)$ of all algebras satisfying SP, and by Orejas and Navarro [21] in the case where the class of algebras is restricted to LOG-algebras.

- We have only considered here the techniques of proofs by consistency. Everone agrees now that they are sometimes inefficient and more direct inductive proof methods have been developed, for instance by U.Reddy [27] in the equational case, or by E.Kounalis and M.Rusinowitch [19] in the Horn clause case. Extending these methods for proving theorems in parameterized specifications does not seem too difficult, and would lead to interesting applications.

Acknowledgements: I sincerely thank Jean-Luc Rémy for his contribution to a first part of this work presented at the COMPASS Workshop in Bremen in 1989, and his useful comments on this version. My thanks also to the anonymous referees for their constructive remarks.

References

[1] L. Bachmair. *Proof methods for equational theories.* PhD thesis, University of Illinois, Urbana-Champaign, 1987. Revised version, August 1988.

[2] L. Bachmair. Proof by consistency in equational theories. In *3rd Symp. Logic in Computer Science*, pages 228–233, Edinburgh, Scotland, 1988. IEEE.

[3] L. Bachmair, N. Dershowitz, and D. Plaisted. Completion without failure. In H. Ait-Kaci and M. Nivat, editors, *Resolution of Equations in Algebraic Structures, Volume 2: Rewriting Techniques*, pages 1–30. Academic Press, 1989.

[4] R. Bundgen. Design, implementation and application of an extended ground reducibility test. Technical Report 88-05, University of Delaware USA, December 1987.

[5] H. Comon. An effective method for handling initial algebras. In P. Lescanne J. Grabowski and W. Wechler, editors, *Proc. Int. Workshop on Algebraic and Logic Programming*, pages 108–118. Akademie Verlag, 1988. Also in Springer-Verlag Lecture Notes in Computer Science, volume 343.

[6] N. Dershowitz. Computing with rewrite systems. *Information and Control*, 65(2/3):122–157, 1985.

[7] L. Duponcheel, L. Jadoul, and W. Van Puymbroeck. Generic proofs by consistency. Technical report, Bell telephone Mfg. Co., Francis Wellesplein 1, B-2018 Antwerp, Belgium, 1989.

[8] H. Ehrig and B. Mahr. *Fundamentals of Algebraic Specification 1. Equations and initial semantics*, volume 6 of *EATCS Monographs on Theorical Computer Science.* Springer-Verlag, 1985.

[9] H. Ganzinger. Parameterized specifications: parameter passing and implementation with respect to observability. *ACM Transactions on Programming Languages and Systems*, 5(3):318–354, 1983.

[10] H. Ganzinger. Ground term confluence in parametric conditional equational specifications. In F. Brandenburg, G. Vidal-Naquet, and M. Wirsing, editors, *Proceedings STACS 87*, volume 247 of *Lecture Notes in Computer Science*, pages 286–298. Springer-Verlag, 1987.

[11] J. Goguen. Parameterized programming. *Transactions on Software Engineering*, SE-10(5):528–543, September 1984.

[12] J. Goguen, J. Meseguer, and D. Plaisted. Programming with parameterized abstract objects in OBJ. In *Theory and Practice of Software Technology*, pages 163–193. North-Holland, 1983.

[13] G. Grätzer. *Universal Algebra.* Springer-Verlag, 1979. Second Edition.

[14] G. Huet and D. Oppen. Equations and rewrite rules: A survey. In R. Book, editor, *Formal Language Theory: Perspectives and Open Problems*, pages 349–405. Academic Press, New York, 1980.

[15] J.-P. Jouannaud and E. Kounalis. Proof by induction in equational theories without constructors. In *Proceedings 1st Symp. on Logic In Computer Science*, pages 358–366, Boston (USA), 1986.

[16] J.-P. Jouannaud and E. Kounalis. Automatic proofs by induction in theories without constructors. *Information and Computation*, 82:1–33, 1989.

[17] D. Kapur, P. Narendran, and H. Zhang. On sufficient completeness and related properties of term rewriting systems. *Acta Informatica*, 24:395–415, 1987.

[18] E. Kounalis. Testing for inductive (co)-reducibility. In A. Arnold, editor, *Proceedings 15th CAAP, Copenhagen (Denmark)*, volume 431 of *Lecture Notes in Computer Science*, pages 221–238. Springer-Verlag, May 1990.

[19] E. Kounalis and M. Rusinowitch. Mechanizing inductive reasoning. In *Proceedings of the AAAI Conference*, pages 240–245, Boston, 1990. AAAI Press and the MIT Press.

[20] G. A. Kucherov. A new quasi-reducibility testing algorithm and its application to proofs by induction. In P. Lescanne J. Grabowski and W. Wechler, editors, *Proc. workshop on Algebraic and Logic Programming*, pages 204–213. Akademie Verlag, 1988. Also in Springer-Verlag Lecture Notes in Computer Science, volume 343.

[21] M. Navarro and F. Orejas. Parameterized Horn clause specifications: proof theory and correctness. In *Proceedings TAPSOFT Conference*, volume 249 of *Lecture Notes in Computer Science*. Springer-Verlag, 1987.

[22] T. Nipkow and G. Weikum. A decidability result about sufficient completeness of axiomatically specified abstract data types. In *6th GI Conference*, volume 145 of *Lecture Notes in Computer Science*, pages 257–268. Springer-Verlag, 1983.

[23] F. Orejas. A characterization of passing compatibility for parameterized specifications. *Theoretical Computer Science*, pages 205–214, 1987.

[24] P. Padawitz. The equational theory of parameterized specifications. *Information and Computation*, 76:121–137, 1988.

[25] D. Plaisted. Semantic confluence and completion method. *Information and Control*, 65:182–215, 1985.

[26] A. Poigné. Parameterisation for order-sorted algebraic specifications. Technical report, Dept. of Computing, Imperial College, London, 1986.

[27] U. Reddy. Term rewriting induction. In M. Stickel, editor, *Proceedings 10th International Conference on Automated Deduction, Kaiserslautern (Germany)*, volume 449 of *Lecture Notes in Computer Science*, pages 162–177. Springer-Verlag, 1990.

[28] J.L. Rémy and H. Zhang. Reveur4 : a system for validating conditional algebraic specifications of abstract data types. In T. O'Shea, editor, *Proceedings of the 5th European Conference on Artificial Intelligence*, Pisa, Italy, 1984. ECAI, Elsevier Science Publishers (North Holland).

[29] W. Taylor. Equational logic. *Houston Journal of Mathematics*, 1979. Appears also in [13], Appendix 4.

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique

ISSN 0249 - 6399