Theoretical Computer Science 517 (2014) 102–142

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

# Parameterisation for abstract structured specifications

Ionuţ Ţuţu [a,b,*]

[a] Department of Computer Science, Royal Holloway University of London, United Kingdom
[b] Institute of Mathematics of the Romanian Academy, Research group of the project ID-3-0439, Romania

ARTICLE INFO

*Article history:*
Received 18 March 2012
Received in revised form 5 November 2013
Accepted 12 November 2013
Communicated by D. Sannella

*Keywords:*
Algebraic specification
Institution theory
Structured institutions
Parameterisation
Free extensions

ABSTRACT

We investigate multiple-parameterised specifications and their instantiation within the institution-independent framework of abstract structured specifications. Our work identifies a set of distinctive features of specifications languages that have a fundamental role in defining and instantiating parameterised specifications. We consider both simultaneous and sequential instantiation of parameters, and allow not only sharing between the body of the parameterised specification and the instances of the parameters, but also between the parameters of a generic specification. The developments conclude with the examination of the relation between the results of simultaneous and sequential instantiation of parameters, which are shown to be isomorphic under a given set of sufficient abstract conditions.



## 1. Introduction

Modularisation plays a paramount role in managing the inherent complexity of large software development projects. This paper is dedicated to the study of parameterisation as one of the most important techniques used in structuring formal specifications. Parameterisation, or generalization, allows abstracting away those elements of a formal specification whose details are not part of the essence of the specified system, and can be obtained at a later time through particularisation.

Parameterised constructions arise naturally in both mathematics and computing science. Immediate examples can be found in the theory of algorithms and data types where most of the entities are parameterised by a combination of structures, operations on structures or properties of them. For instance, one can easily see that the list structure is generic with respect to the type of its elements. Most programming or specification languages allow the definition of such data types in a manner that expresses clearly their variable components. In the case of the list structure we identify a single parameter, namely the type of its elements. The particularisations of a generic entity are obtained by instantiating its parameters. This requires fitting argument mappings for each of the parameters meant to be instantiated. For the actual situation of generic lists, we can choose to instantiate the type of list's elements, which makes no assumptions on their structure, simply by replacing it with the type of natural numbers, thus obtaining lists of natural numbers. In conclusion, parameterisation as a technical device has a double function; it allows the writer to make explicit the parameters of a generic entity and at the same time it provides a suitable mechanism for instantiating the parameters.

From the point of view of formal specifications, parameterisation has an essential function in increasing the expressive power of the specification languages that support this mechanism. When certain conditions are met by the base specification language, the systematic use of parameterisation allows the development of complex module expressions in which new parameterised specifications can be obtained by partially instantiating existing generic specifications, or instantiating their parameters with other generic specifications. As it was pointed out in [16] we can gain in this way both the specification power of richer languages and the desired properties for specification and verification of the simpler ones.

* Correspondence to: Department of Computer Science, Royal Holloway University of London, United Kingdom.
*E-mail address:* ittutu@gmail.com.

0304-3975/$ – see front matter 
http://dx.doi.org/10.1016/j.tcs.2013.11.008

### 1.1. The structure and the contents of the paper

Our paper presents an overview of the theory of parameterised specifications in the abstract framework of institution theory [18]. Its goal is to establish those properties of specification languages with a primary function in building parameterised specifications and in supporting the instantiation of the parameters such that the resulting specifications satisfy expected algebraic properties. To this effect, our work upgrades the theory of parameterised specifications introduced in [13] in two fundamental ways:

1. We consider the general case of multiple-parameterised specifications in which the parameters can interact between them and also with any additional specifications involved in the instantiation process – as opposed to the original approach, which required parameters to have distinct signatures, disjoint from the signatures of other parameters or of the possible instances of the parameters.
2. The investigations follow the recent developments in the field of abstract structured specifications [10], and in this way are independent not only of the underlying logical system but also of the actual structuring operators. Consequently, the constructions and the results discussed here can be applied uniformly to specification frameworks that are based on either model-oriented [28,13] or property-oriented [12] studies of modularisation.

These upgrades bring a significant increase in the flexibility of parameterised specifications, which is required by the way that parameterisation is used in practice. For instance, given the following CASL [26] specification of weighted lists, in order to obtain, by instantiation, the specification WEIGHTED_LIST [NAT] of weighted lists of natural numbers, one has to accept the sharing of the specification of natural numbers by the parameter of the generic specification WEIGHTED_LIST and its instance NAT. This is achieved in our example by treating the specification NAT as an import of the parameterised specification, i.e. as an auxiliary specification that is used by the parameter and is not meant to be instantiated.

**spec** NAT =
  **free type** $Nat ::= 0 \mid s\_(Nat)$
**end**

**spec** NAT_PLUS =
  NAT
**then op** $\_ + \_ : Nat \times Nat \to Nat$
  $\forall M, N : Nat$
  • $0 + N = M$
  • $s\ M + N = s\ (M + N)$
**end**

**spec** WEIGHTED_LIST [**sort** *Elt* **op** $weight : Elt \to Nat$] **given** NAT =
  NAT_PLUS
**then free type** $List ::= nil \mid \_\_\_\_(Elt; List)$
  **op** $weight : List \to Nat$
  $\forall E : Elt; L : List$
  • $weight(nil) = 0$
  • $weight(E\ L) = weight(E) + weight(L)$
**end**

Note that in this situation the sharing between the parameters and their instances is explicitly stated at design time through imports. Moreover, the imports of any generic CASL specification are fixed and common to all the parameters, to their instances and also to the body of the parameterised specification. From this perspective, what is distinctive about the approach discussed in the present paper is that the sharing between the parameters and their instances, or between the body of the parameterised specification and the instances of the parameters, is not fixed, but is decided at the time of the instantiation.

Another relevant example is given by the CafeOBJ [11] specification of generic pairs listed below. The parameters of PAIR are isomorphic renamings ELT.*FIRST* and ELT.*SECOND* of the specification ELT and have the signatures given by the sorts *Elt.FIRST* and *Elt.SECOND*, respectively. For technical reasons, unless explicitly stated otherwise, the CafeOBJ specifications implicitly protect the predefined specification BOOL of boolean values. Therefore, the symbols defined by BOOL such as *true* and *false* are shared between the parameters ELT.*FIRST* and ELT.*SECOND*, and thus it is natural to consider that sharing may also occur between the parameters of a generic specification. Although this particular situation is supported by the current implementation of the system through module sharing, the semantics of CafeOBJ requires disjoint signatures for any two different parameters of a given parameterised module. The subsequent technical sections of the paper will present a number of more elaborated examples.