# Object Specification Logic

Amílcar Sernadas, Cristina Sernadas
IST, Department of Mathematics, Lisbon, Portugal
José Félix Costa
Faculdade de Ciências, Department of Informatics, Lisbon, Portugal

November 2, 1995

**Abstract**

A logic for specifying and reasoning about object classes and their instances (aspects) is presented and illustrated. This logic is an extension of a rather standard linear temporal, many-sorted, first-order predicate logic with equality. The extensions where designed to be as simple as possible while supporting the envisaged locality of arguments, object specialization and object aggregation. Objects are specified through their aspects. Each aspect establishes a local vocabulary (signature). The logic works at two levels: first, we can specify and prove assertions about a given object aspect in isolation (local reasoning), eg persons, or patients, or cars; second, we can specify interaction constraints and make inferences between aspects within the same community of objects (global reasoning), eg carry the theorems of persons onto patients (specialization inheritance), or carry the theorems of persons onto the aggregations of persons and cars (incorporation inheritance). Some reflection mechanisms are also shown to be sound: for instance what becomes true of persons because of the interactions between persons and cars. The proposed logic is given in an axiomatic style.

Key words: System specification and verification, object-orientation, logic of inheritance, logic of interaction, temporal logic, many-sorted logic.

## 1 Introduction

Object-oriented programming has become increasingly popular thanks to the expected advantages in software development and maintenance productivity [Bro90, Cox90]. Such advantages are presented as corollaries of the intensive use of the data encapsulation and reuse facilities made available by the object-oriented development languages and systems [DMN67, GR83, Str86, Cox86, Mey88]. Object-orientation seems also to pay-off at the level of software specification and design [Boo86, SFSE89a, Nie89]. Taking for granted this thesis, one should ask what kind of object-oriented specification language should be adopted. Several have been proposed in the literature [SFSE89a, SE91, JSHS91, SRGS91].

We defend a formal approach to specification since the price of lower flexibility at the early stages is well counterbalanced by the advantages of a rigourous semantics and correctness verifiability. The latter imposes more than a formal specification language: a deductive system is needed also, making it a specification/verification logic. Indeed, object-orientation promises a lot at the level of the deductive system: it should be possible to carry out "local" proofs and later on assess if and how the resulting theorems carry over to the "whole".

Actually, we defend a temporal logic approach to object specification, in the sequel of much work in the general direction of using temporal logic for (information) system specification triggered by [Pnu77, Ser80] and later explored for instance in [FS88, SFSE89b, CS91, FSMS91, FM92]. Compared with [FM92] our approach puts an emphasis on object classes (and their instances) and deals with interaction in a different way. Instead of using theory morphisms and colimit constructions for describing interacting objects, we express all constraints (including interaction ones) as formulae (of the same logic but possibly in different vocabulary contexts). Morphisms are used only to set-up the relationships between aspects (eg specialization). This approach seems to be more amenable to extension to classes.

In a nutshell, we propose and illustrate the use of a logic of "object aspects". This logic is an extension of a rather standard linear temporal, many-sorted, first-order predicate logic with equality. The extensions where designed to be as simple as possible while supporting the envisaged locality of arguments, object specialization and object aggregation. Objects are specified through their aspects. Each aspect establishes a local vocabulary (signature). The logic works at two levels: first, we can specify and prove assertions about a given object aspect in isolation (local reasoning), eg persons, or patients, or cars; second, we can specify interaction constraints and make inferences between aspects within the same community of objects (global reasoning), eg carry the theorems of persons onto patients (specialization inheritance), or carry the theorems of persons onto the aggregations of persons and cars (incorporation inheritance). Some reflection mechanisms are also shown to be sound: for instance what becomes true of persons because of the interactions between persons and cars.

The proposed logic is given in an axiomatic style. We decided to refrain from considering in this paper more practical versions, for instance sequent-based as in [SGL92] or tableau-based as in [LS91, GSGA93] where only local reasoning has been considered. Also, we only establish soundness. It should be possible to obtain a (relative) completeness result but that is a topic for further research. The proposed logic is monotonic, although we recognize that a full account of object specialization (with overriding) will need non-monotonic features [BL91, CSS94].

We assume that the reader is conversant with the field of temporal logic specification (for instance at the level of [McA76, Gol87]). For an overview see [Eme90]. We also use a little bit of category theory (the reader may find all relevant concepts in the introductory chapters of any textbook on the matter, eg [AHS90]): the more exotic notion of freely generated cartesian category is reviewed in the Appendix (for more details see [Sza78]). We also assume that the reader is familiar with the use of many-sorted formalisms for specification purposes, either in the algebraic abstract data type setting (see [EM85]), or in the program correctness context (see [TZ88]). For a recent account of many sorted logic and its applications see [MT93].

Before entering the presentation of the logic, Section 2 discusses the underlying object framework (following [ES91]) and outlines the main requirements for a logic of aspects. The local logic of object aspects is established in Section 3: language, semantics and deductive system. Section 4 presents the auxiliary notions of aspect signature morphism and aspect specification morphism, needed in the sequel. Finally, Section 5 presents the global logic for reasoning about a community of objects, including specialization and aggregation without and with interaction: language, semantics and deductive system. Note that the usefulness of some peculiar features of the local logic given in Section 3 only become clear in Section 5.

## 2 Object and aspects

At first sight, an object is an entity with an internal state (reflected in the values of its slots or attributes) that may interact with other objects. While so interacting an object displays some behaviour. That is, depending on its internal state, it is not always ready to provide the same services to the others and/or to ask the same services of others. The most basic unit of interaction is the event: for instance, the event open-door may be shared by a person and a car. Therefore, the behaviour of an object may be seen as corresponding to a state-dependent menu of enabled actions (at each state the set of actions in which the object is "willing" to involve itself).

At this naive level of analysis, already proposed in [SFSE89b], the specification of an object should include the list of its attributes (with the indication of their codomain sorts), the list of its actions (with the indication of their parameter sorts if any) and the constraints on its behaviour.

As an illustration consider books as objects. Among their attributes we may find: code (taking values in the set of natural numbers), registered-in (taking values in the set of books), and available (taking boolean values). For instance, available=true is supposed to mean that a book is available for borrowing. If so, in such a state of the book under consideration the observation available(true) is possible. Among the actions of books we may consider: is-taken (with a parameter taking values in the set of users) and is-returned (with a similar parameter). For instance, the occurrence of the action is-taken(John) is supposed to mean that the book under consideration was at that point taken out by user John. Clearly, that action may only happen in a state where the book is available.

Actually, the picture is a little bit more complex: what we have been talking about so far is just an object aspect ([ES91]). Indeed, if we consider an object like "Abstract and Concrete Categories" in some library, it may have several aspects: it is a book (having all properties of books in that library), but it may also be a reserved-book (having the added properties of the so called reserved books that may not be taken away from the library). The other objects may interact with it as a book, disregarding the additional services of a reserved book if any, unknowing that its behaviour may be constrained because it is also a reserved book. Of course, they may also interact with it as a reserved book. In short, "Abstract and Concrete Categories" may play the role of a book although it is a reserved book. Indeed, it might even play other roles like thing. This mechanism (polymorphism by inclusion) is essential to reuse, one of the key features of object-orientation.

Therefore, an object is, in general, a collection of aspects consistent with each other. From a specification point of view, it is clear that we should specify only object aspects! That is, objects are specified through their aspects. Therefore, we need an object aspect specification logic for "local" reasoning about a given aspect disregarding the other aspects.

But the interplay between the different aspects of the same object is to be carried out at the "global" level where we recognize relationships between different aspects: for instance, the specialization relationship we have been illustrating. Furthermore, at the global level we can also impose the interactions that may exist between different aspects: for instance, when a book is-taken we know that a user takes it. Thus, interaction constraints can be expressed by formulae relating events in different aspects.

To this end, it is important to consider aggregation aspects, such as the composite of a book and an user where the interactions between that book and that user are to be specified. Then, the same specification and verification techniques can be used for specifying local and global (interaction) behaviour.

Note also that similar aspects constitute aspect types. It makes more sense to specify aspect types instead of aspects per se. Moreover, we may need attributes that take values in aspect types (beside data types). For illustration consider the aspect type book composed of all books, including "Abstract and Concrete Categories".

It is important to stress that we are dealing here with aspect types and not object types (see [ES91]). Indeed, each object may have several related aspects (recall the example of "Abstract and Concrete Categories" with two aspects: a book aspect and a reserved-book aspect). Each of its aspects is an instance of an aspect type. Specialization appears at the aspect type level as follows: for instance we say that reserved-book is a subtype of book, or book is a supertype of reserved book. Aggregation is reflected at the type level also. We need aggregation aspect types, such as book-user whose instances are the pairs of books and users.

En passant note that associated to each aspect type we may have an aspect class, or in the traditional but misleading object-oriented terminology, an object class. A class is more than a type: it is itself an aspect with an attribute listing all the existing instances of that aspect type.

Therefore, the envisaged aspect specification logic must provide the means for referring to individuals of different types (eg integer, book, user, book-user, reserved-book), hence the many-sorted first-order approach. This approach should be ordered for aspect types so that the specialization sub-supertype relationships can be made explicit (this idea is also explored in [BHPS93]). The specification logic must also provide the means for describing the behaviour of each individual, hence the temporal approach. The signature of each aspect should include its attributes and actions. Both are introduced as (state-dependent) predicates following [FM92].

The proposed aspect specification logic includes some generic axioms that reflect universal properties of aspects, namely: functional constraint on attribute predicates; sequentiality of prime (non-aggregation) aspects; frame constraint (locality as in [FM92]).

The properties of data types (eg boolean, integer) are assumed to given once and for all, in order to concentrate on the specification and verification of object aspects.

With respect to the interplay between the local and global levels of reasoning, we should expect some form of inheritance mechanisms, eg for specialization (allowing the inclusion of theorems about a supertype into the set of theorems about a subtype). Less expected are some reflection mechanisms that allow the theorems to be obtained in an aspect by taking into account its embedding in the community of aspects, eg for aggregation (if we know that every user returns the books that he/she takes away, we can conclude that every book is returned).

# 3 Object aspect specification

## 3.1 Local language

Recall that a data signature is a pair $\Sigma = \langle S, OP\rangle$ where $S$ is a set and $OP$ is a $S^* \times S$-indexed family of sets. The elements of $S$ are said to be *sorts* and, for each $(w, s) \in S^* \times S$, $OP_{w,s}$ is the set of *operation symbols* with parameter list $w$ and result sort $s$. Given an $S$-indexed family $Z$ of sets of variables, for each $s \in S$, $T_{\Sigma,s}(Z)$ denotes the set of terms of sort $s$ that may be inductively built with the variables in $Z$ over $\Sigma$. An *equation* over a signature $\Sigma$ is a triple $\langle Z, t, t'\rangle$ where $Z$ is an $S$-indexed family of sets and $t, t' \in T_{\Sigma,s}(Z)$ for some $s \in S$. A *data specification* is a pair $spec = \langle \Sigma, Eq\rangle$ where $\Sigma$ is a data signature and $Eq$ is a set of equations over $\Sigma$.

From now on, we assume as given a fixed data specification $spec_{DT} = \langle \Sigma_{DT}, Eq_{DT}\rangle$ with $\Sigma_{DT} = \langle S_{DT}, OP_{DT}\rangle$. This data specification is called the *data environment*.

In the examples below we assume that the data signature $\Sigma_{DT}$ contains interalia: $S_{DT} = \{bool, nat\}$, $(OP_{DT})_{\epsilon,bool} = \{true, false\}$ and $(OP_{DT})_{bool,bool} = \{not\}$.

We are now ready to introduce the notion of aspect signature corresponding to a given class of objects. Clearly, it should include the attribute observation symbols and the action symbols of that class. Moreover, since attributes/actions may point to and have parameters in instances of other classes, we need to include the context of aspect sorts that are visible. Among these aspect sorts, we find the underlying aspect sort of the individuals of the class at hand. The visible aspect sorts are ordered for depicting the specialization relationship between them (for a similar approach see [BHPS93]). Finally, we must also consider all possible aggregations of the prime visible aspect sorts. To this end we use a canonical construction described in [Sza78] and summarized in the Appendix. Note that this construction provides symbols for denoting the projections from an aggregation sort into each of its components.

**Definition 3.1** An *aspect signature* is a tuple $\Phi = \langle S_{GS,\leq}, \alpha, OBS, ACT \rangle$ where:

- $S_{GS,\leq}$ is the cartesian category freely generated from a partial order $\langle GS, \leq \rangle$ where $GS$ is a finite set;
- $\alpha$ is an element of $AS = |S_{GS,\leq}|$;
- $OBS$ is an $(S_{DT} \times AS)^{+}$-indexed family of finite sets;
- $ACT$ is an $(S_{DT} \times AS)^{*}$-indexed family of finite sets.

In this context, the elements of $AS$ are said to be *aspect sorts*; the elements of $GS$ are called *generator or prime sorts*; each product $s_1 \times ... \times s_n$ is said to be the *aggregation sort* of $s_1, ..., s_n$ with *sort projections* $\pi_1 : s_1 \times ... \times s_n \rightarrow s_1, ..., \pi_n : s_1 \times ... \times s_n \rightarrow s_n$; for each pair $s, s' \in GS$ such that $s \leq s'$, $\iota : s \rightarrow s'$ is said to be the *sort inclusion* of $s$ into $s'$ (the identity in $s$ when $s' = s$); the terminal sort is denoted by 1; the partial order $\leq$ on prime sorts is called the *inheritance relation*: whenever $s \leq s'$ we say that $s$ is a *subsort* of $s'$ and $s'$ is a *supersort* of $s$; $\alpha$ is the *underlying sort*; each $OBS_{w\ s}$ is the set of *attribute observation symbols* with parameter sort list $w$ and result sort $s$; and each $ACT_w$ is the set of *action symbols* with parameter sort list $w$. If $\alpha \in GS$ then $\Phi$ is said to be a *prime aspect signature*. Otherwise, $\Phi$ is called a *composite aspect signature*.

For illustration purposes consider a library where for the moment we want to have only books, users and reserved books. Assume the following object aspect signature $\Phi_{book}$ for books:

- $GS_{book} = \{book, user\}$;
- $\leq_{book} = id_{GS_{book}}$;
- $\alpha_{book} = book$;
- $(OBS_{book})_{bool} = \{available\}$;
- $(OBS_{book})_{nat} = \{code\}$;
- $(OBS_{book})_{book} = \{registeredin\}$;
- $(ACT_{book})_{nat\ book} = \{\star\}$;
- $(ACT_{book})_{user} = \{istaken, isreturned\}$.

Consider the following object aspect signature $\Phi_{user}$ for users:

- $GS_{user} = \{book, user\}$;
- $\leq_{user} = id_{GS_{user}}$;
- $\alpha_{user} = user$;

- $(OBS_{user})_{book\ bool} = \{borrowedbook\}$;
- $(ACT_{user})_{\epsilon} = \star$;
- $(ACT_{user})_{book} = \{takes, returns\}$.

Finally, consider the following object aspect signature $\Phi_{resbook}$ for reserved-books (whose borrowing may be suspended):

- $GS_{resbook} = \{resbook, book, user\}$;
- $\leq_{resbook} = id_{GS_{resbook}} \cup \{\langle resbook, book\rangle\}$;
- $\alpha_{resbook} = resbook$;
- $(OBS_{resbook})_{bool} = \{available, suspended\}$;
- $(OBS_{resbook})_{nat} = \{code\}$;
- $(OBS_{resbook})_{book} = \{registeredin\}$;
- $(ACT_{resbook})_{nat\ book} = \{\star\}$;
- $(ACT_{resbook})_{user} = \{istaken, isreturned\}$;
- $(ACT_{resbook})_{\epsilon} = \{issuspended, isresumed\}$.

Given an aspect signature, we proceed to introduce the terms and formulae that we can write over that signature. First, we recognize the available predicate symbols.

**Definition 3.2** Given an aspect signature $\Phi = \langle S_{GS,\leq}, \alpha, OBS, ACT\rangle$, $PRD$ is the $(SDT \times AS)^*$-indexed family of sets of *predicate symbols* such that:

- $PRD_{\epsilon} = \{occc : c \in ACT_{\epsilon}\}$;
- $PRD_w = \{occc : c \in ACT_w\} \cup \{o : o \in OBS_w\}$ for each $w \in (S_{DT} \times AS)^+$.

The predicate *occc* refers to the actual occurrence (happening) of $c$. We refrain to consider here the enabling predicate *enbc* indicating when the action $c$ is enabled as proposed in [FS88]. Indeed, such a distinction would imply the adoption of a more complicated semantics (at least multilinear). We return to this issue in the concluding remarks.

Terms are built using the usual constructions found in any many-sorted formalism (eg [TZ88]). Moreover, terms of aspect sort $s'$ can be built from terms of aspect sort $s$ using any morphism $\nu : s \rightarrow s'$.

**Definition 3.3** Let $\Phi = \langle S_{GS,\leq}, \alpha, OBS, ACT\rangle$ be an aspect signature and $Z$ an $(S_{DT} \times AS)$-indexed family of sets of variables. The $(S_{DT} \times AS)$-indexed family $T_{\Phi}(Z)$ of sets of *terms* is inductively built as follows using the family $Z$ of variables with the operation symbols in $OP_{DT}$ and the sort morphisms in $S_{GS,\leq}$:

- $z \in T_{\Phi,s}(Z)$ provided that $z \in Z_s$;
- $h(t_1, ..., t_n) \in T_{\Phi,s}(Z)$ provided that $h \in OP_{DT_{s_1 \ldots s_n,s}}$ and each $t_i \in T_{\Phi,s_i}(Z)$;
- $\nu(t) \in T_{\Phi,s'}(Z)$ provided that $\nu : s \rightarrow s'$ and $t \in T_{\Phi,s}(Z)$.

Concerning formulae, we found useful to distinguish between (aspect) schema formulae and aspect (instance) formulae. The latter are needed when the reference to the instance at hand is important, eg in interactions as we shall see later on. The schema language is almost the standard many-sorted, linear temporal first-order with equality language within the context at hand of aspect signatures. For the temporal constructs see for instance [Gol87]: as usual, $\mathsf{X}$ is the next operator, $\mathsf{F}$ is the sometime operator, $\mathsf{G}$ is the always operator and $\mathsf{U}$ is the until operator. Note only the unary predicates allowing the recognition that a term of a given supersort is also of a given subsort. These will be needed when reflecting theorems from the subtype into the supertype, as we shall see later.

**Definition 3.4** Let $\Phi = \langle S_{GS,\leq}, \alpha, OBS, ACT \rangle$ be an aspect signature and $Z$ an $(S_{DT} \times AS)$-indexed family of sets of variables. The many-sorted, first-order, linear temporal *schema language* with equality $SL_\Phi(Z)$ is inductively built as follows using the predicate symbols in $PRD$ and the terms in $T_\Phi(Z)$:

1. $(t_1 = t_2) \in SL_\Phi(Z)$ provided that $t_1, t_2 \in T_{\Phi,s}(Z)$ for some sort $s$;
2. $s(t) \in SL_\Phi(Z)$ provided that $t \in T_{\Phi,s'}(Z)$ and $s \leq s'$;
3. $p \in SL_\Phi(Z)$ provided that $p \in PRD_\epsilon$;
4. $p(t_1, ..., t_n) \in SL_\Phi(Z)$ provided that $p \in PRD_{s_1 \ldots s_n}$ and each $t_i \in T_{\Phi,s_i}(Z)$;
5. $(\neg f) \in SL_\Phi(Z)$ provided that $f \in SL_\Phi(Z)$;
6. $(f \wedge g), (f \vee g), (f|g), (f \Rightarrow g), (f \Leftrightarrow g) \in SL_\Phi(Z)$ provided that $f, g \in SL_\Phi(Z)$;
7. $(\forall x f) \in SL_\Phi(Z)$ provided that $f \in SL_\Phi(Z)$ and $x \in Z_s$ for some sort $s$;
8. $(\mathsf{X} f), (\mathsf{F} f) \in SL_\Phi(Z)$ provided that $f \in SL_\Phi(Z)$;
9. $(f \,\mathsf{U}\, g) \in SL_\Phi(Z)$ provided that $f, g \in SL_\Phi(Z)$.

Basically, an aspect (instance) formula is a schema formula instanciated on a given variable of the underlying aspect sort.

**Definition 3.5** Given an aspect signature $\Phi = \langle S_{GS,\leq}, \alpha, OBS, ACT \rangle$, each element of the aspect language $L_\Phi$ is a pair $(Z, z(f))$ where:

- $Z$ is an $(S_{DT} \times AS)$-indexed family of sets of variables;
- $z \in Z_\alpha$;
- $f \in SL_\Phi(Z)$;
- $f$ does not contain any quantification on variable $z$;
- $z(f)$ is obtained from $f$ by uniformly replacing each symbol $p \in PRD$ by $z.p$.

In this context, we say that $z$ is the *target variable* of the formula.

The following expressions are abbreviated formulae (second element $z(f)$ of a formula $(Z, z(f))$ of $L_{\Phi_{book}}$:

- $(b.occ\star(n, b') \Rightarrow (\mathsf{X}\, b.available(true)))$;
- $(b.occistaken(u) \Rightarrow (\mathsf{X}\, b.available(false)))$;
- $(b.occistaken(u) \Rightarrow b.available(true))$.

The variables are mnemonic of their sorts ($b, b' : book, u : user, n : nat$). For instance, the last of these formulae states that if the action $istaken(u)$ occurs in the instance $b$ of book then the observation $available(true)$ holds in that same instance $b$. The notion of aspect specification as a theory presentation is as usual (see for instance [GB92]).

**Definition 3.6** An *aspect specification* is a pair $spec = \langle \Phi, Ax \rangle$ where $\Phi$ is an object aspect signature and $Ax \subseteq L_\Phi$. In this context, the elements of Ax are called (local) *aspect axioms.*

**Example 3.7** Consider the aspect specification

$$spec_{book} = \langle \Phi_{book}, Ax_{book} \rangle$$

where $Ax_{book}$ includes the following axioms:

1. $(b.occ\star(n, b') \Rightarrow (\mathsf{X}(\mathsf{G}(\neg\, b.occ\star(n', b'')))))$;
2. $(b.occ\star(n, b') \Rightarrow (\mathsf{X}\, b.available(true)))$;
3. $(b.occ\star(n, b') \Rightarrow (\mathsf{X}(\mathsf{G}\, b.code(n))))$;
4. $(b.occ\star(n, b') \Rightarrow (\mathsf{X}(\mathsf{G}\, b.registeredin(b'))))$;
5. $(b.occistaken(u) \Rightarrow b.available(true))$;
6. $(b.occisreturned(u) \Rightarrow b.available(false))$;
7. $(b.occistaken(u) \Rightarrow (\mathsf{X}(b.available(false)))$;
8. $(b.occisreturned(u) \Rightarrow (\mathsf{X}(b.available(true)))$.

Axiom 1 states that no $\star$-action of $b$ may occur after a $\star$-action of $b$. Axiom 4 imposes that the book $b'$ where $b$ is registered is indicated by the second parameter of the $birth$-action of $b$. Axiom 5 states that if the action $istaken(u)$ occurs then $available(true)$ must hold. Finally, axiom 7 states the effects of the occurrence of action $istaken(u)$: immediately afterwards the value of the attribute $available$ becomes $true$.

**Example 3.8** Consider the aspect specification

$$spec_{user} = \langle \Phi_{user}, Ax_{user} \rangle$$

where $Ax_{user}$ includes the following axioms:

1. $(u.occ\star \Rightarrow (\mathsf{X}(\mathsf{G}(\neg\, u.\star))))$;
2. $(u.occ\star \Rightarrow (\mathsf{X}(\neg(\exists b\ u.borrowedbook(b, true)))))$;
3. $(u.occtakes(b) \Rightarrow u.borrowedbook(b, false))$;
4. $(u.occreturns(b) \Rightarrow u.borrowedbook(b, true))$;
5. $(u.occtakes(b) \Rightarrow (\mathsf{X}\, u.borrowedbook(b, true)))$;
6. $(u.occreturns(b) \Rightarrow (\mathsf{X}\, u.borrowedbook(b, false)))$;
7. $(u.occtakes(b) \Rightarrow (u.borrowedbook(b, true)\, \mathsf{U}\, u.occreturns(b)))$.

Axiom 7 states that the occurrence of action $takes(b)$ implies that book $b$ is borrowed until the occurrence of the action $returns(b)$.

**Example 3.9** Consider the aspect specification

$$spec_{reservedbook} = \langle \Phi_{reservedbook}, Ax_{reservedbook} \rangle$$

where $Ax_{reservedbook}$ includes the following axioms:

1. $(r.occ\star(n, b) \Rightarrow (\mathsf{X}\, r.suspended(false)))$;
2. $(r.occistaken(u) \Rightarrow r.suspended(false))$;
3. $(r.occissuspended \Rightarrow (r.available(true) \wedge r.suspended(false)))$;
4. $(r.occisresumed \Rightarrow r.suspended(true))$;
5. $(r.occissuspended \Rightarrow (\mathsf{X}\, r.suspended(true)))$;
6. $(r.occisresumed \Rightarrow (\mathsf{X}\, r.suspended(false)))$;
7. $(\neg(r.occissuspended \wedge r.occistaken(u)))$;
8. $(\mathsf{X}(\mathsf{F}\, r.occissuspended))$.

Clearly, we also expect to bring in the properties of book. Such a mechanism for inheritance into subtypes is an essential piece of the envisaged deductive system (see 5.3 below).

In the sequel we work with the following basic sublanguage $BSL_\Phi(Z)$:

1. $(t_1 = t_2) \in BSL_\Phi(Z)$ provided that $t_1, t_2 \in T_{\Phi,s}(Z)$ for some sort $s$;
2. $p \in BSL_\Phi(Z)$ provided that $p \in PRD_\epsilon$;
3. $p(t_1, ..., t_n) \in BSL_\Phi(Z)$ provided that $p \in PRD_{s_1 \ldots s_n}$ and each $t_i \in T_{\Phi,s_i}(Z)$;
4. $(\neg f) \in BSL_\Phi(Z)$ provided that $f \in BSL_\Phi(Z)$;
5. $(f \Rightarrow g) \in BSL_\Phi(Z)$ provided that $f, g \in BSL_\Phi(Z)$;
6. $(\forall x f) \in BSL_\Phi(Z)$ provided that $f \in BSL_\Phi(Z)$ and $x \in Z_s$ for some sort $s$;
7. $(\mathsf{X}\, f), (\mathsf{G}\, f) \in BSL_\Phi(Z)$ provided that $f \in BSL_\Phi(Z)$;
8. $(f \,\mathsf{U}\, g) \in BSL_\Phi(Z)$ provided that $f, g \in BSL_\Phi(Z)$.

We recover the full $SL_\Phi(Z)$ with the standard abbreviations: $(f \vee g)$ is $((\neg f) \Rightarrow g)$; $(f \wedge g)$ is $(\neg(f \Rightarrow (\neg g)))$; $(f|g)$ is $((f \wedge (\neg g)) \vee ((\neg f) \wedge g))$; $(f \Leftrightarrow g)$ is $((f \Rightarrow g) \wedge (g \Rightarrow f))$; and, finally, $(\mathsf{F}\, f)$ is $(\neg(\mathsf{G}(\neg f)))$.

## 3.2 Local semantics

We assume as given some fixed $spec_{DT}$-algebra $dt$ that we call *data environment algebra*, or, simply, *data algebra* (see [EM85]). For each data sort $s \in S_{DT}$, $dt_s$ is the carrier set of sort $s$. For each operation symbol $h \in (OP_{DT})_{w,s}$, $[\![h]\!]_{dt}$ is the interpretation given by the algebra $dt$ to $h$.

We are now ready to introduce the notion of interpretation structure for a given aspect signature. It is composed of two parts. The first part provides the carriers for the aspect sorts, generalizing the usual many-sorted family of carriers to the aggregation sorts and subsorts. In the sequel $Set$ denotes the category of sets and total maps. The second part is a linear Kripke structure, traditional for linear temporal logic semantics (see [Gol87]) with some necessary adaptations and constraints.

**Definition 3.10** Given an aspect signature $\Phi = \langle S_{GS,\leq}, \alpha, OBS, ACT \rangle$, an *interpretation structure* over $\Phi$ is a pair $asp = \langle U, \lambda \rangle$ where:

- $U : S_{GS,\leq} \rightarrow Set$ is a cartesian functor such that $U(s) \subseteq U(s')$ whenever $s \leq s'$ (in the sequel we may write $U(s)$ instead of $dt_s$ for each $s \in S_{DT}$);
- $\lambda : \mathbb{N} \rightarrow IB$ where $IB = \wp\{u.p(v_1, ..., v_n) : u \in U(\alpha), p \in PRD_{s_1,...,s_n}, v_i \in U(s_i)$ for $i = 1, ..., n$ with $n \geq 0\}$ such that
    1. if $\{u.o(v_1, ..., v_n) \in \lambda_{k+1}\} \neq \{u.o(v_1, ..., v_n) \in \lambda_k\}$ then there is an action symbol $c$ such that $u.occc(v_1, ..., v_n) \in \lambda_k$ for suitable $v_1, ..., v_n$;
    2. for each observation symbol $o$, if $u.o(v_1, ..., v_n), u.o(v'_1, ..., v'_n) \in \lambda_k$ and $v_i = v'_i$ for $i = 1, ..., n-1$ then $v_n = v'_n$;
    3. for each action symbol $c$, if $u.occo(v_1, ..., v_n), u.occo(v'_1, ..., v'_n) \in \lambda_k$ then $v_i = v'_i$ for $i = 1, ..., n$.

Clause (1) states that changes in the set of observations are made only by actions: similar to the locality concept in [FM92]. Clause (2) states that attributes are functional. Clause (3) states the mutual exclusion of actions of the same kind, that is, it imposes the sequentiality of each basic component of the object aspect.

We may want to extract from an interpretation structure the life-cycle (sequence of observations and action occurences) of a particular instance of the underlying aspect sort, as follows:

**Definition 3.11** Let $asp = \langle U, \lambda \rangle$ be an interpretation structure over the signature $\Phi = \langle S_{GS,\leq}, a, OBS, ACT \rangle$. For each $u \in U(\alpha)$, we extract from $\lambda$ the sequence (called the *life cycle of instance u*) $\lambda_u : \mathbb{N} \rightarrow SB$ where
$SB = \wp\{p(v_1, ..., v_n) : p \in PRD_{s_1,...,s_n}, v_i \in U(s_i)$ for $i = 1, ..., n$ with $n \geq 0\}$
such that
$\lambda_{u,k} = \{o(v_1, ..., v_n) : u.o(v_1, ..., v_n) \in \lambda_k\} \cup \{occc(v_1, ..., v_n) : u.occc(v_1, ..., v_n) \in \lambda_k\}$.

We proceed now to establish the interpretation of terms and satisfaction of formulae, after introducing the necessary notion of assignment of values to variables. These definitions are quite standard in a many-sorted, linear temporal, first-order environment, barring some peculiarities raised by the relationships between aspect sorts. For the temporal operators see for instance [Gol87]. For the many-sorted issues see for instance [End72, Man93].

**Definition 3.12** Given an interpretation structure $asp = \langle U, \lambda \rangle$ over the signature $\Phi = \langle S_{GS,\leq}, \alpha, OBS, ACT \rangle$ and an $(S_{DT} \times AS)$-indexed family $Z$ of variables, an *assignment* to $Z$ is an $(S_{DT} \times AS)$-indexed family $asg$ of maps such that $asg_s : Z_s \rightarrow U(s)$.

**Definition 3.13** Given an interpretation structure $asp = \langle U, \lambda \rangle$ over the signature $\Phi = \langle S_{GS,\leq}, \alpha, OBS, ACT \rangle$, an $(S_{DT} \times AS)$-indexed family $Z$ of variables and an assignment $asg$ to $Z$, the *interpretation* over $asp$ of terms in $T_\Phi(Z)$ is defined as follows:

- $[\![z]\!]_{asp,asg} = asg_s(z)$ for $z \in Z_s$;
- $[\![h(t_1, ..., t_n)]\!]_{asp,asg} = [\![h]\!]_{dt}([\![t_1]\!]_{asp,asg}, ..., [\![t_n]\!]_{asp,asg})$ for $h \in OP_{DT_{s_1,...,s_n}}$;
- $[\![\nu(t)]\!]_{asp,asg} = U(\nu)([\![t]\!]_{asp,asg})$ for $t \in T_{\Phi,s}(Z)$, for each $\nu : s \rightarrow s'$.

The interpretation of terms built with sort morphisms is defined using the interpretation (a total map in $Set$) of the morphism given by the functor $U$.

**Definition 3.14** Given an interpretation structure $asp = \langle U, \lambda \rangle$ over the signature $\Phi = \langle S_{GS,\leq}, a, OBS, ACT \rangle$, an $(S_{DT} \times AS)$-indexed family $Z$ of variables, an assignment $asg$ to $Z$ and a position $k \in I\!N$, the *satisfaction* by $asp$ *for* asg *at* $k$ is inductively defined for all formulae on $Z$ with target $z$ as follows:

- $asp \vDash_{asg,k} (Z, (t_1 = t_2))$ iff $[\![t_1]\!]_{asp,asg} = [\![t_2]\!]_{asp,asg}$;
- $asp \vDash_{asg,k} (Z, s(t))$ iff $[\![t]\!]_{asp,asg} \in U(s)$;
- $asp \vDash_{asg,k} (Z, z.p)$ iff $asg(z).p \in \lambda_k$ for $p \in PRD_\epsilon$;
- $asp \vDash_{asg,k} (Z, z.p(t_1, ..., t_n))$
  iff $asg(z).p([\![t_1]\!]_{asp,asg}, ..., [\![t_n]\!]_{asp,asg}) \in \lambda_k$ for $p \in PRD_{s_1,...,s_n}$;
- $asp \vDash_{asg,k} (Z, z(\neg f))$ iff not $asp \vDash_{asg,k} (Z, z(f))$;
- $asp \vDash_{asg,k} (Z, z(f \Rightarrow g))$ iff $asp \vDash_{asg,k} (Z, z(f))$ implies $asp \vDash_{asg,k} (Z, z(g))$;
- $asp \vDash_{asg,k} (Z, z(\forall x f))$
  iff $asp \vDash_{asg',k} (Z, z(f))$ for every assignment $asg'$ $x$-equivalent to $asg$;
- $asp \vDash_{asg,k} (Z, z(\mathsf{X} f))$ iff $asp \vDash_{asg,k+1} (Z, z(f))$;
- $asp \vDash_{asg,k} (Z, z(\mathsf{G} f))$ iff $asp \vDash_{asg,k'} (Z, z(f))$ for every $k' \geq k$;
- $asp \vDash_{asg,k} (Z, z(f \mathsf{U} g))$
  iff $asp \vDash_{asg,k} (Z, z(g))$ for some $k' \geq k$ and
  $asp \vDash_{asg,k} (Z, z(f))$ for every $k''$ such that $k \leq k'' \leq k'$.

**Definition 3.15** Let $asp = \langle U, \lambda \rangle$ be an interpretation structure over the signature $\Phi = \langle S_{GS,\leq}, a, OBS, ACT \rangle$. We say that *asp satisfies a formula* $(Z, z(f)) \in L_\Phi$, written $asp \vDash (Z, z(f))$, iff $asp \vDash_{asg,k} (Z, z(f))$ for every assignment $asg$ to $Z$ and every position $k \in I\!N$. We say that *asp satisfies a specification spec* $= \langle \Phi, Ax \rangle$, written $asp \vDash spec$ iff $asp \vDash (Z, z(f))$ for every formula $(Z, z(f)) \in Ax$.

Validity and semantic closure are as expected: a formula is valid iff it is satisfied by every interpretation structure; the semantic closure of a specification includes all formulae that are satisfied by all interpretation structures that satisfy the axioms (see for instance [GB92]).

**Definition 3.16** Let $\Phi = \langle S_{GS,\leq}, \alpha, OBS, ACT \rangle$ be an aspect signature. We say that a formula $(Z, z(f)) \in L_\Phi$ is *valid*, written $\vDash_\Phi (Z, z(f))$, iff $asp \vDash (Z, z(f))$ for every interpretation structure $asp$ over $\Phi$.

**Definition 3.17** Let $spec = \langle \Phi, Ax \rangle$ be a specification. The *semantic closure* of *spec* is the following set of formulae in $L_\Phi$:
$Ax^{\vDash_\Phi} = \{(Z, z(f)) : asp \vDash (Z, z(f))$ for each interpretation structure $asp$ such that $asp \vDash spec\}$.
Whenever $(Z, z(f)) \in Ax^{\vDash_\Phi}$ we write $\vDash_{spec} (Z, z(f))$.

We also need the following notion of semantic entailment within the context of a given specification.

**Definition 3.18** Let $spec = \langle \Phi, Ax \rangle$ be a specification and $H$ a set of formulae of $L_\Phi$. The set of formulae *semantically entailed* by $H$ within *spec* is as follows:
$Ax^{H \vDash_\Phi} = \{(Z, z(f)) : asp \vDash (Z, z(f))$ for each interpretation structure $asp$ such that $asp \vDash (\Phi, Ax \cup H)$.
Whenever $(Z, z(f)) \in Ax^{H \vDash_\Phi}$ we write $H \vDash_{spec} (Z, z(f))$.

It is not worthwhile to illustrate these concepts at this stage since they have their counterparts in the deductive system.

### 3.3 Local reasoning

Our objective at this stage is to introduce a deductive system for reasoning about object aspects in isolation, that is not taking into account the community where they may be inserted. The deductive system must include the usual axioms for a many-sorted, first-order, linear temporal logic with equality (see [End72, Gol87]) plus some specific axioms related to the general properties of object aspects. The inference rules should be the traditional ones.

We adopt the following conventions: $(\forall f), (\exists f)$ stand for the universal, existential closure of $f$, respectively; $\underline{x}, \underline{x'}, \underline{x_1}, ..., \underline{x_n}$ are suitable tuples of variables; equality between tuples stand for conjunction of equalities between the corresponding pairs of variables.

**Definition 3.19** Let $spec = \langle \Phi, Ax \rangle$ be a specification. The *theory generated* by $spec$ is the least set of formulae of $L_\Phi$, denoted $Ax^{\vdash_\Phi}$, satisfying the following requirements:

1. $(Z, z(f)) \in Ax^{\vdash_\Phi}$
   provided that $f$ is a tautological formula: $TautA$;
2. $(Z, z((\forall x f) \Rightarrow g)) \in Ax^{\vdash_\Phi}$
   provided that $g$ is is obtained from $f$ by uniformly replacing all free occurrences of $x$ by $t$ where $t$ is free for $x$ in $f$: $QA1$;
3. $(Z, z((\forall x(f \Rightarrow g)) \Rightarrow ((\forall x f) \Rightarrow (\forall x g)))) \in Ax^{\vdash_\Phi}$: $QA2$;
4. $(Z, z(f \Rightarrow (\forall x f))) \in Ax^{\vdash_\Phi}$
   provided that $x$ does not occur free in $f$: $QA3$;
5. $(Z, (x = x)) \in Ax^{\vdash_\Phi}$: $RefA$;
6. $(Z, ((x = y) \Rightarrow z(f \Rightarrow g))) \in Ax^{\vdash_\Phi}$
   where $f, g$ are atomic and $g$ is obtained from $f$ by replacing zero or more occurrences of $x$ by $y$: $CongA$;
7. $(Z, z((\mathsf{G}(f \Rightarrow g)) \Rightarrow ((\mathsf{G}\, f) \Rightarrow (\mathsf{G}\, g)))) \in Ax^{\vdash_\Phi}$: $\mathsf{G}\, A$;
8. $(Z, z((\mathsf{X}(f \Rightarrow g)) \Rightarrow ((\mathsf{X}\, f) \Rightarrow (\mathsf{X}\, g)))) \in Ax^{\vdash_\Phi}$: $\mathsf{X}\, A$;
9. $(Z, z((\mathsf{X}(\neg f)) \Leftrightarrow (\neg(\mathsf{X}\, f)))) \in Ax^{\vdash_\Phi}$: $\mathsf{X}\, NA$;
10. $(Z, z((\mathsf{G}\, f) \Rightarrow (f \wedge (\mathsf{X}(\mathsf{G}\, f))))) \in Ax^{\vdash_\Phi}$: $\mathsf{G}\,\mathsf{X}\, A$;
11. $(Z, z((\mathsf{G}(f \Rightarrow (\mathsf{X}\, f))) \Rightarrow (f \Rightarrow (\mathsf{G}\, f)))) \in Ax^{\vdash_\Phi}$: $IndA$;
12. $(Z, z((f \,\mathsf{U}\, g) \Rightarrow (\mathsf{F}\, g))) \in Ax^{\vdash_\Phi}$: $UFA$;
13. $(Z, z((f \,\mathsf{U}\, g) \Rightarrow (g \vee (f \wedge (\mathsf{X}(f \,\mathsf{U}\, g)))))) \in Ax^{\vdash_\Phi}$:$\mathsf{U}\,\mathsf{X}\, A$;
14. $(Z, z((\forall x(\mathsf{G}\, f)) \Rightarrow (\mathsf{G}(\forall x f)))) \in Ax^{\vdash_\Phi}$: $\mathsf{G}\, BarcA$;
15. $(Z, z((\forall x(\mathsf{X}\, f)) \Rightarrow (\mathsf{X}(\forall x f)))) \in Ax^{\vdash_\Phi}$: $\mathsf{X}\, BarcA$;
16. $Ax \subseteq Ax^{\vdash_\Phi}$: $ASA$;
17. $(Z, (\forall (t = t'))) \in Ax^{\vdash_\Phi}$
   provided that the data sort terms $t$ and $t'$ have the same value in $dt$ for every assignment: $PDtA$;
18. $(Z, (\neg(\forall (t = t')))) \in Ax^{\vdash_\Phi}$
   provided that the data sort terms $t$ and $t'$ do not have the same value in $dt$ for some assignment: $NDtA$;
19. $x.(\mu_n(...(\mu_1(y)...) = \mu(y)) \in Ax^{\vdash_c}$
   provided that $\mu = \mu_n \circ ... \circ \mu_1$: $MorA$;

20. $(Z, z((\exists(\neg(o(\underline{x}) \Leftrightarrow (\mathsf{X}\, o(\underline{x}))))) \Rightarrow ((\exists occc_1(\underline{x}_1)) \vee ... \vee occc_n(\underline{x}_n))))) \in Ax^{\vdash_\Phi}$
for each observation symbol symbol $o$, assuming that $\{c_1, ..., c_n\}$ is the set of all action symbols of $\Phi$: $FrmA_o$;

21. $(Z, z((o(\underline{x}, x) \wedge o(\underline{x}, x')) \Rightarrow (x = x'))) \in Ax^{\vdash_\Phi}$
for each observation symbol $o$: $FuncA_o$;

22. $(Z, z((occc(\underline{x}) \wedge occc(\underline{x'})) \Rightarrow (\underline{x} = \underline{x'}))) \in Ax^{\vdash_\Phi}$
for each action symbol $c$: $ExcA_c$;

23. If $(Z, z(f)), (Z, z(f \Rightarrow g)) \in Ax^{\vdash_\Phi}$ then $(Z, z(g)) \in Ax^{\vdash_\Phi}$: $DetR$;

24. If $(Z, z(f)) \in Ax^{\vdash_\Phi}$ then $(Z, z(\forall x f)) \in Ax^{\vdash_\Phi}$: $GenR$;

25. If $(Z, z(f)) \in Ax^{\vdash_\Phi}$ then $(Z, z(\mathsf{G}\, f)) \in Ax^{\vdash_\Phi}$: $\mathsf{G}\, NecR$;

26. If $(Z, z(f)) \in Ax^{\vdash_\Phi}$ then $(Z, z(\mathsf{X}\, f)) \in Ax^{\vdash_\Phi}$: $\mathsf{X}\, NecR$.

The specific axioms concerning the underlying data structure ($PDtA, NDtA$) reflect for the sake of simplicity all assertions valid in dt. We adopt this high handed approach because we do not want to concern ourselves with the reasoning about data values. The axioms related to sort morphisms $MorA$ are straightforward. The frame axiom schema ($FrmA_o$) reflects the fact that observations may be modified only by actions. This axiom plays the role of the locality rule in [FM92]. The functionality axiom schema ($FuncA_o$) reflects the functionality of attributes. The exclusion axiom schema ($ExcA_c$) reflects the mutual exclusion of similar actions.

The notions of theorem and derivation follow as expected.

**Definition 3.20** Let $spec = \langle \Phi, Ax \rangle$ be a specification. Each element $(Z, z(f))$ of $Ax^{\vdash_\Phi}$ is said to be a *theorem* of *spec*, written $\vdash_{spec} (Z, z(f))$.

**Definition 3.21** Let $spec = \langle \Phi, Ax \rangle$ be a specification and $H \subseteq L_\Phi$. The set of formulae *derived* from $H$ within *spec* is the least set of formulae of $L_\Phi$, denoted $Ax^{H\vdash_\Phi}$, satisfying the following requirements:

- $Ax^{\vdash_\Phi} \subseteq Ax^{H\vdash_\Phi}$;
- $H \subseteq Ax^{H\vdash_\Phi}$;
- If $(Z, z(f)), (Z, z(f \Rightarrow g)) \in Ax^{H\vdash_\Phi}$ then $(Z, z(g)) \in Ax^{H\vdash_\Phi}$: $DetR$.

We write $H \vdash_{spec} (Z, z(f))$ whenever $(Z, z(f)) \in Ax^{H\vdash_\Phi}$.

It is straightforward to establish by induction the soundness of the proposed deductive system:

**Theorem 3.22** (Soundness Theorem)
Let $spec = \langle \Phi, Ax \rangle$ be a specification, $H \subseteq L_\Phi$ and $(Z, z(f)) \in L_\Phi$. If $H \vdash_{spec} (Z, z(f))$ then $H \vDash_{spec} (Z, z(f))$.

Note the careful usage of the inference rules (only detachment is used when making derivations) so that we can obtain the metatheorems in their simplest formulation. Indeed:

**Theorem 3.23** Let $spec = \langle \Phi, Ax \rangle$ be a specification and $(Z, z(f)), (Z', z'(f')) \in L_\Phi$. Then:

- If $(Z, z(f)) \vdash_{spec} (Z', z'(f')))$ then $\vdash_{spec} ((Z, z(f)) \Rightarrow (Z', z'(f')))$: $DED$;

- If $(Z, z(f)) \vdash_{spec} (Z', z'(f'))$ and $(Z, z(f)) \vdash_{spec} (Z', z'(\neg f'))$ then $\vdash_{spec} (Z, z(\neg f))$: $ABS$.

**Example 3.24**

$$\vdash_{spec_{book}} (\neg((\exists u\ b.occistaken(u)) \wedge (\exists u\ b.occisreturned(u))))$$

Derivation:

| | | |
|---|---|---|
| 0. | $((\exists u\ b.occistaken(u)) \wedge (\exists u.\ b.occisreturned(u)))$ | $Hyp$ |
| 1. | $(\exists u\ b.occistaken(u))$ | $TautA, DetR: 0$ |
| 2. | $(b.occistaken(u) \Rightarrow b.available(true))$ | $ASA$ |
| 3. | $((\exists u\ b.occistaken(u)) \Rightarrow b.available(true))$ | $FOT, DetR: 2$ |
| 4. | $(\exists u\ b.occisreturned(u))$ | $TautA, DetR: 0$ |
| 5. | $(b.occisreturned(u) \Rightarrow b.available(false))$ | $ASA$ |
| 6. | $((\exists u\ b.occisreturned(u)) \Rightarrow b.available(false))$ | $FOT, DetR: 5$ |
| 7. | $(b.available(true) \wedge b.available(false))$ | $TautA, DetR: 1, 3, 4, 6$ |
| 8. | $((b.available(x) \wedge b.available(x')) \Rightarrow (x = x'))$ | $FuncA_{available}$ |
| 9. | $((b.available(true) \wedge b.available(false)) \Rightarrow (true = false))$ | $QA1, DetR: 8$ |
| 10. | $(true = false)$ | $DetR: 7, 9$ |
| 11. | $(\neg(true = false))$ | $NDtA$ |
| 12. | $(\neg((\exists u\ b.occistaken(u)) \wedge (\exists u\ b.occisreturned(u))))$ | $ABS: 0, 10, 11$ |

By $FOT$ we mean a well known theorem of the first-order predicate calculus.

**Example 3.25**

$$\vdash_{spec_{resbook}} (\mathsf{X}(\mathsf{F}\ r.available(true)))$$

Derivation:

| | | |
|---|---|---|
| 0. | $(\mathsf{X}(\mathsf{F}\ r.occissuspended))$ | $ASA$ |
| 1. | $(r.occissuspended \Rightarrow (r.available(true) \wedge r.suspended(false)))$ | $ASA$ |
| 2. | $(r.occissuspended \Rightarrow r.available(true))$ | $TauTA, DetR: 1$ |
| 3. | $(\mathsf{X}(\mathsf{F}\ r.available(true)))$ | $TLT, DetR: 0, 2$ |

By $TLT$ we mean a well known theorem of the linear temporal calculus.

The latter result although trivial will be used later on for illustrating the reflection mechanism of the global deductive system.

# 4 Object aspect morphisms

In order to explain inheritance and aggregation we need to be able to relate aspect specifications. This objective is achieved by introducing the notion of aspect specification morphism. But first we have to consider aspect signature morphisms.

## 4.1 Signature morphism

An aspect signature morphism should map aspect sorts into aspect sorts, observation symbols into observation symbols and action symbols into action symbols, preserving the (transformed) parameter sorts (as already described in [FM92] capitalizing in the tradition of many-sorted formalisms [EM85]).

Furthermore, we must be careful concerning the cartesian category of aspect sorts: its structure (aggregation sorts and subsorts) should be preserved by the signature morphism. Finally, note that we should allow the change of the underlying sort not according to the sort map. For instance, in the inclusion morphism from the aspect signature of book into the aspect signature of reserved-book, the aspect sort *book* is mapped onto itself by the sort map, but we want the underlying aspect sort to be *resbook* in the latter signature! However, we should be able to relate the instances of the latter underlying sort with the instances of the former underlying sort. In the example at hand, the sort inclusion morphism from *resbook* into *book* does the trick. The usage of this subtle mechanism for aggregation will be illustrated later on.

**Definition 4.1** An *aspect signature morphism*

$$\sigma : \langle S_{GS,\leq}, \alpha, OBS, ACT \rangle \rightarrow \langle S_{GS',\leq'}, \alpha', OBS', ACT' \rangle$$

is a quadruple $\langle \sigma_{AS}, \sigma_\alpha, \sigma_{OBS}, \sigma_{ACT} \rangle$ where

- $\sigma_{AS} : S_{GS,\leq} \rightarrow S_{GS',\leq'}$ is a cartesian functor such that $\sigma_{AS}(s) \leq' \sigma_{AS}(s')$ whenever $s \leq s'$;
- $\sigma_\alpha : \alpha' \rightarrow \sigma_{AS}(\alpha)$ is a morphism of $S_{GS',\leq'}$;
- $\sigma_{OBS}$ is an $(S_{DT} \times AS)^+$-indexed family of maps: each map is $\sigma_{OBS,w} : OBS_w \rightarrow OBS'_{\sigma^+_{AS}}(w)$;
- $\sigma_{ACT}$ is an $(S_{DT} \times AS)^*$-indexed family of maps: each map is $\sigma_{ACT,w} : ACT_w \rightarrow ACT'_{\sigma^*_{AS}}(w)$.

The aspect signature morphism is said to be an *inclusion* when the functor $\sigma_{AS}$ is an inclusion, $\sigma_\alpha$ is a sort inclusion and the maps $\sigma_{OBS}, \sigma_{ACT}$ are inclusions. The aspect signature morphism is said to be an *injection* when the functor $\sigma_{AS}$ is an embedding, $\sigma_\alpha$ is a sort projection and the maps $\sigma_{OBS}, \sigma_{ACT}$ are injections.

We have an inclusion signature morphism $\eta : \Phi_{book} \rightarrow \Phi_{resbook}$ from the signature of book into the signature of reserved book and injection signature morphisms $\rho_i$ : $\Phi_{book} \rightarrow \Phi_{book \times book}$, i=1,2 from book and book into the aggregation of the two books.

Given a signature morphism $\sigma : \Phi \rightarrow \Phi'$, we should be able to translate formulae of $\Phi$ into formulae of $\Phi'$. Clearly, observation and action symbols must be replaced according to their maps of the signature morphism, as already used in [FM92].

However, we must be careful with the sorts of the variables. All variables change sorts according to the sort map of the signature morphism, with the exception of the target variable since this must become of the underlying sort of the aspect signature $\Phi'$. Therefore, in order to keep sort consistency, we must replace occurrences of the target variable $z$ in parameters by the term of the correct sort $\sigma_\alpha(z)$.

**Definition 4.2** Given an aspect signature morphism $\sigma : \Phi \rightarrow \Phi'$, the *translation* of a formula $(Z, z(f)) \in L_\Phi$ by $\sigma$ is the formula $(Z_\sigma, z(\sigma(f))) \in L_{\Phi'}$ such that:

- $\sigma(f) \in SL_{\Phi'}(Z_\sigma)$ is obtained from the schema $f$ by replacing each $o \in (OBS_\Phi)_w$ by $\sigma_{OBS,w}(o)$, each $c \in (ACT_\Phi)_w$ by $\sigma_{ACT,w}(c)$ and $z$ by the term $\sigma_\alpha(z)$;
- $Z_\sigma$ is the $(S_{DT} \times AS')$-indexed family of sets of variables defined as follows: $x \in (Z_\sigma)_{s'}$ whenever either $x$ is not $z$ and $x \in Z_s$ and $s' = \sigma_{AS}(s)$, or $x$ is $z$ and $s' = \alpha'$.

Consider the inclusion aspect signature morphism $\eta : \Phi_{book} \rightarrow \Phi_{resbook}$. The translation of the book axiom

$$(b.occistaken(u) \Rightarrow b.available(true))$$

is

$$(b.occistaken(u) \Rightarrow b.available(true))$$

but where $b$ becomes of sort $resbook$. The translation of

$$(b.occ\star(n, b') \Rightarrow (\mathsf{X}(\mathsf{G}\, b.registeredin(b'))))$$

is the formula

$$(b.occ\star(n, b') \Rightarrow (\mathsf{X}(\mathsf{G}\, b.registeredin(b'))))$$

where $b$ is of sort $resbook$ but $b'$ remains of sort $book$. The translation of

$$b.registeredin(b)$$

is

$$b.registeredin(\iota_{book,resbook}(b)).$$

Consider the injection aspect signature morphism $\rho : \Phi_{book} \rightarrow \Phi_{book \times user}$. Assuming that $\rho$ is an inclusion on observations and actions, the translation of

$$(b.occistaken(u) \Rightarrow b.available(true))$$

is

$$(b.occistaken(u) \Rightarrow b.available(true))$$

but where $b$ becomes of sort $book \times user$. The translation of

$$(b.occ\star(n, b') \Rightarrow (\mathsf{X}(\mathsf{G}\, b.registeredin(b'))))$$

is

$$(b.occ\star(n, b') \Rightarrow (\mathsf{X}(\mathsf{G}\, b.registeredin(b'))))$$

where $b$ is of sort $book \times user$ but $b'$ remains of sort $book$. And

$$(b.registeredin(b))$$

is translated onto

$$(b.registeredin(\pi_1(b))).$$

Consider an injection aspect signature morphism $\rho : \Phi_{book} \rightarrow \Phi_{book \times book}$. The translation of

$$(b.occistaken(u) \Rightarrow b.available(true))$$

is

$$(b.occ\rho_{ACT}(istaken(u)) \Rightarrow b.\rho_{OBS}(available(true)))$$

with $b$ of sort $book \times book$.

## 4.2 Specification morphisms: specialization and incorporation

The notion of specification morphism follows as usual (see [GB92]). We also identify two special cases: one useful when describing specialization and the other useful for dealing with aggregation.

**Definition 4.3** An *aspect specification morphism* $\sigma : \langle \Phi, Ax \rangle \to \langle \Phi', Ax' \rangle$ is an aspect signature morphism $\sigma : \Phi \to \Phi'$ such that
$\sigma(AX) = \{(Z_\sigma, z(\sigma(f))) : (Z, z(f)) \in AX\} \subseteq (Ax')^{\models_{\Phi'}}$.

**Definition 4.4** An aspect specification morphism $\sigma : \langle \Phi, Ax \rangle \to \langle \Phi', Ax' \rangle$ is said to be a *specialization* iff the underlying signature morphism is an inclusion.

Specialization morphisms are important to explain inheritance. We want the inclusion aspect signature morphism $\eta : book \to resbook$ to induce a specialization morphism. Thus, the translation of every axiom on books is to be entailed by the specification of reserved books.

**Definition 4.5** An aspect specification morphism $\sigma : \langle \Phi, Ax \rangle \to \langle \Phi', Ax' \rangle$ is said to be an *incorporation* iff the underlying signature morphism is an injection.

Incorporation morphisms are important to explain the relationship between an aggregation and its components. We want the injection aspect signature morphism $\rho_i : \Phi_{book} \to \Phi_{book \times book}$, for i=1,2, to induce an incorporation morphism. Thus, the translation of every axiom on books is to be entailed (twice) by the specification of $book \times book$.

## 4.3 The category of aspect specifications

Having aspect signature and aspect specification morphisms we can define a category of aspect signatures and a category of aspect specifications. In both cases we need coproducts as we shall see in the next Section.

**Theorem 4.6** Aspect signatures and their aspect signature morphisms constitute a category: *ASig*.

**Theorem 4.7** The category *ASig* is cocartesian.

Proof:

- The aspect signature $\langle S_{\emptyset,\emptyset}, 1, \emptyset, \emptyset \rangle$ is initial;
- The coproduct $\langle S_{GS,\leq}, \alpha, OBS, ACT \rangle$
of $\langle S_{GS',\leq'}, \alpha', OBS', ACT' \rangle$ and $\langle S_{GS'',\leq''}, \alpha'', OBS'', ACT'' \rangle$ is as follows:
    - $GS = GS' + GS''$ with injections $i'$ and $i''$;
    - $s \leq r$ providing that either $s \leq' r$ when $s, r \in GS'$ or $s \leq'' r$ when $s, r \in GS''$;
    - $\alpha$ is $i'_e(\alpha') \times i''_e(\alpha'')$ where $i'_e, i''_e$ are the canonical extensions of $i', i''$ to $AS'$ and $AS''$, respectively;
    - $OBS$ is the $(S_{DT} \times AS)^{+}$-indexed family of the least sets such that if $o' \in OBS'_w$ then $j'(o') \in OBS_{i'^{+}_x(w)}$ and if $o'' \in OBS''_w$ then $j''(o'') \in OBS_{i''^{+}_x(w)}$, where $i'_x, i''_x$ are the extensions of $i'_e, i''_e$ to $(S_{DT} \times AS')$ and $(S_{DT} \times AS'')$, respectively, such that $i'_x(s) = i''_x(s) = i_{DT}(s)$ for every $s \in s_{DT}$;

– $ACT$ is the $(S_{DT} \times AS)^*$-indexed family obtained in a similar way from the families $ACT'$ and $ACT''$.

**Theorem 4.8** Aspect specifications and their aspect specification morphisms constitute a category: *ASpec*.

**Theorem 4.9** The category *ASpec* is cocartesian.

Proof:

- The aspect specification $\langle\langle S_{\emptyset,\emptyset}, 1, \emptyset, \emptyset\rangle, \emptyset\rangle$ is initial;
- The coproduct $\langle\langle S_{GS,\leq}, \alpha, OBS, ACT\rangle, Ax\rangle$ of $\langle\langle S_{GS',\leq'}, \alpha', OBS', ACT'\rangle, Ax'\rangle$ and $\langle\langle S_{GS'',\leq''}, \alpha'', OBS'', ACT''\rangle, Ax''\rangle$ is as follows:
  - $\langle S_{GS,\leq}, \alpha, OBS, ACT\rangle$ is the coproduct of $\langle S_{GS',\leq'}, \alpha', OBS', ACT'\rangle$ and $\langle S_{GS'',\leq''}, \alpha'', OBS'', ACT''\rangle$ with the signature injections $\sigma'$ and $\sigma''$;
  - $Ax = \sigma'(Ax') \cup \sigma''(Ax'')$.

We might expect the cocompleteness of these categories but since we only need coproducts in the sequel we refrain from considering colimits in general herein.

# 5 Object community specification

We want to define the concept of community specification. For this purpose we must start by introducing a global language.

## 5.1 Global language

The alphabet of the language for specifying communities of aspects should include the relevant aspect signatures with constraints reflecting the specialization and incorporation relationships.

**Definition 5.1** A *community signature* is a pair $\Psi = \langle S_{GS,\leq}, \{\Phi^r\}_{r\in AS}\rangle$ where:

- $S_{GS}, \leq$ is the cartesian category freely generated from a partial order $(GS, \leq)$ where $GS$ is a finite set;
- $AS = |S_{GS}, \leq|$;
- for each $r \in GS$, $\Phi^r = \langle S_{GS^r,\leq^r}, \alpha^r, OBS^r, ACT^r\rangle$ is an aspect signature such that
  - $S_{GS^r,\leq^r}$ is a cartesian full subcategory of $S_{GS,\leq}$;
  - whenever $r \leq r'$, there is an inclusion aspect signature morphism $\eta_{r',r} : \Phi^{r'} \to \Phi^r$;
  - $\Phi^1 = \langle S_{\emptyset,\emptyset}, 1, \emptyset, \emptyset\rangle$ is the initial aspect signature;
  - for each aggregation sort $r = r_1 \times ... \times r_n$ with $r_1, ..., r_n \in GS$, $\Phi^r$ is the coproduct of $\Phi^{r_1}, ..., \Phi^{r_n}$ endowed with injections $\mu_1 : \Phi^{r_1} \to \Phi^r, ..., \mu_n : \Phi^{r_n} \to \Phi^r$ (called the *prime injection morphisms* of the aggregation).

That is, a community signature is a family of consistent aspect signatures in the sense that the required specialization and incorporation morphisms are present.

As an illustration consider the following community signature $\Psi_{library}$:

- $GS_{library} = \{book, user, resbook, librarian\}$;
- $\leq_{library} = id_{GS_{library}} \cup \{\langle resbook, book\rangle\}$;
- $\Phi^{book}_{library}, \Phi^{user}_{library}, \Phi^{resbook}_{library}$ are as given before;
- $\Phi^{librarian}_{library}$ is not detailed here.

A community specification is a community signature endowed with axioms for each of its aspect sorts.

**Definition 5.2** A *community specification* is a pair $cspec = \langle \Psi, \{Ax^r\}_{r\in AS}\rangle$ with $\Psi = \langle S_{GS,\leq}, \{\Phi^r\}_{r\in AS}\rangle$ such that:

- $spec^r = \langle \Phi^r, Ax^r\rangle$ constitutes an aspect specification for each $r \in AS$;
- $Ax^1 = \emptyset$.

**Example 5.3** As an illustration, consider the following specification of the interaction between books and users using formulae of $L_{\Phi^{book\times user}}$:

1. $z.occtakes(\pi_1(z)) \Leftrightarrow z.occistaken(\pi_2(z))$;
2. $z.occreturns(\pi_1(z)) \Leftrightarrow z.occisreturned(\pi_2(z))$.

Axiom 1 imposes that the occurrence of the action $takes(\pi_1(z))$ in the user $\pi_2(z)$ implies the occurrence of the action $istaken(\pi_2(z))$ in the book $\pi_1(z)$, and vice-versa.

Clearly, the specifier must be careful when adding interaction axioms (on the aggregation sorts). The examples above are typical. Also typical are mutual exclusion constraints. The decision on which kinds of interaction axioms to allow is a methodological one.

For instance, one should expect the formula

$$z.occtakes(\pi_1(z)) \Leftrightarrow z.occistaken(\pi_3(z))$$

to be entailed by $Ax^{book\times librarian\times user}$. Indeed, the interaction between books and users should carry through to the aggregation aspect of books, users and librarians together. Moreover, one should expect the formulae

$$z.occ\mu_2(takes)(\pi_1(z)) \Leftrightarrow z.occistaken(\pi_2(z))$$

and

$$z.occ\mu_3(takes)(\pi_1(z)) \Leftrightarrow z.occistaken(\pi_3(z))$$

to be entailed by $Ax^{book\times user\times user}$.

## 5.2 Global semantics

In the global semantics we must state how the local (aspect) interpretation structures are related. For this purpose we need to know what is $\sigma(B)$ for any $B \in SB^{r_1}$ whenever $\sigma : \Phi^{r_1} \to \Phi^{r_2}$ is either an inclusion or an injection aspect signature morphism:

$$\sigma(B) = \{\sigma(p)(v_1, ..., v_n) : p(v_1, ..., v_n) \in B\}$$

where $\sigma(occe) = occ\sigma(e)$.

**Definition 5.4** Given a community signature $\Psi = \langle S_{GS,\leq}, \{\Phi^r\}_{r\in AS}\rangle$ where for each $r \in AS$ $\Phi^r = \langle S_{GS^r,\leq^r}, \alpha^r, OBS^r, ACT^r\rangle$, an *interpretation structure* over $\Psi$ is a pair $com = \langle cU, \{asp^r\}_{r\in AS}\rangle$ where:

- $cU : S_{GS,\leq} \rightarrow Set$ is a cartesian functor such that $cU(s) \subseteq cU(s')$ whenever $s \leq s'$ (in the sequel, we may write $cU(s)$ instead of $dt_s$ for each $s \in S_{DT}$);
- each $asp^r = \langle U^r, \lambda^r \rangle$ is an interpretation structure for $\Phi^r$ such that
  1. $U^r(s) = cU(s)$ for each $s \in AS^r$, $U^r(\nu) = cU(\nu)$ for every $\nu \in Mor_{S_{GS^r},\leq^r}(s, s')$;
  2. $(\lambda^r_{u,k}) \downarrow_{\eta_{r',r}(SB^{r'})} = \eta_{r',r}(\lambda^{r'}_{cU(\eta_{r',r_\alpha})(u),k})$, whenever $r \leq r'$;
  3. $(\lambda^{r_1 \times \ldots \times r_n}_{u,k}) \downarrow_{\mu_i(SB^{r_i})} = \mu_i(\lambda^{r_i}_{cU(\mu_{i_\alpha})(u),k})$ for $i = 1, ..., n$.

Clause (2) states that whenever $r \leq r'$ we obtain from the life cycle of an instance $u$ of sort $r$ the image of the life cycle of the instance $u$ of sort $r'$ when we only keep at each point predicates in the image of $SB^{r'}$. Clause (3) states a similar constraint for aggregations.

We now proceed to establish the notions of interpretation of terms, satisfaction of formulae, validity, semantic closure, and semantic entailment within the context of a community interpretation structure. These definitions parallel what we did before for aspects, with the necessary adaptations, namely concerning the indexing on aspect signatures throughout.

**Definition 5.5** Given an interpretation structure $com = \langle cU, \{asp^r\}_{r \in AS} \rangle$ over the signature $\Psi = \langle S_{GS,\leq}, \{\Phi^r\}_{r \in AS} \rangle$ and an $(S_{DT} \times AS)$-indexed family $Z$ of variables, an *assignment* to $Z$ is an $(S_{DT} \times AS)$-indexed family $asg$ of maps such that $asg_s : Z_s \rightarrow cU(s)$.

**Definition 5.6** Given an interpretation structure $com = \langle cU, \{asp^r\}_{r \in AS} \rangle$ over the signature $\Psi = \langle S_{GS,\leq}, \{\Phi^r\}_{r \in AS} \rangle$, an $(S_{DT} \times AS)$-indexed family $Z$ of variables and an assignment $asg$ to $Z$, the *restriction* $asg^r$ of $asg$ to $\Phi^r$ is an $(S_{DT} \times AS^r)$-indexed family of maps such that $(asg^r)_s = asg_s$ for each $s \in S_{DT} \times AS^r$.

**Definition 5.7** Given an interpretation structure $com = \langle cU, \{asp^r\}_{r \in AS} \rangle$ over the signature $\Psi = \langle S_{GS,\leq}, \{\Phi^r\}_{r \in AS} \rangle$, an $(S_{DT} \times AS)$-indexed family $Z$ of variables and an assignment $asg$ to $Z$, the *interpretation over com through asg* of terms in $T_{\Phi^r}(Z)$ is defined as follows: $[\![t]\!]_{com,asg} = [\![t]\!]_{asp^r,asg^r}$.

**Definition 5.8** Given an interpretation structure $com = \langle cU, \{asp^r\}_{r \in AS} \rangle$ over the signature $\Psi = \langle S_{GS,\leq}, \{\Phi^r\}_{r \in AS} \rangle$, an $(S_{DT} \times AS)$-indexed family $Z$ of variables, an assignment $asg$ to $Z$, a formula $(Z, z(f)) \in L_{\Phi^r}$ and $k \in I\!N$, *com satisfies* $(Z, z(f))$ for *assignment asg* at *position k*, written $com \vDash_{asg,k} (Z, z(f))$, iff $asp^r \vDash_{asg^r,k} (Z, z(f))$.

**Definition 5.9** Let $com = \langle cU, \{asp^r\}_{r \in AS} \rangle$ be an interpretation structure over the signature $\Psi = \langle S_{GS,\leq}, \{\Phi^r\}_{r \in AS} \rangle$. We say that *com satisfies a formula* $(Z, z(f)) \in L_{\Phi^r}$, written $com \vDash (Z, z(f))$, iff $com \vDash_{asg,k} (Z, z(f))$ for every assignment $asg$ to $Z$ and every position $k \in I\!N$. We say that *com satisfies a specification* $cspec = \langle \Psi, \{Ax^r\}_{r \in AS} \rangle$, written $com \vDash cspec$, iff $com \vDash (Z, z(f))$ for every formula $(Z, z(f))$ of $Ax^r$ for every $r \in AS$.

**Definition 5.10** Let $\Psi = \langle S_{GS,\leq}, \{\Phi^r\}_{r \in AS} \rangle$ be a community signature. We say that a formula $(Z, z(f)) \in L_{\Phi^r}$ is *valid*, written $\vDash_\Psi (Z, z(f))$, iff $com \vDash (Z, z(f))$ for every interpretation structure $com$ over $\Psi$.

**Definition 5.11** Let $cspec = \langle \Psi, \{Ax^r\}_{r \in AS} \rangle$ be a community specification with signature $\Psi = \langle S_{GS,\leq}, \{\Phi^r\}_{r \in AS} \rangle$. The *semantic closure* of $cspec$ is the family $Ax^{\vDash_\Psi} = \{Ax^{\vDash_{\Psi^r}}\}_{r \in AS}$ such that each $Ax^{\vDash_{\Psi^r}}$ is the following subset of the language $L_{\Phi^r}$:
$\{(Z, z(f)) : com \vDash (Z, z(f))$ for each $\Psi$ interpretation $com$ such that $com \vDash cspec\}$.
Whenever $(Z, z(f)) \in Ax^{\vDash_{\Psi^r}}$ we write $\vDash_{cspec^r} (Z, z(f))$.

**Definition 5.12** Let $cspec = \langle \Psi, \{Ax^r\}_{r \in AS} \rangle$ be a community specification with signature $\Psi = \langle S_{GS,\leq}, \{\Phi^r\}_{r \in AS} \rangle$ and $H = \{H^r\}_{r \in AS}$ where each $H^r$ is a set of formulae of $L_{\Phi^r}$. Each element $Ax^{H \models_{\Psi^r}} \subseteq L_{\Phi^r}$ of the $AS$-indexed family $Ax^{H \models_\Phi}$ of sets of formulae *semantically entailed* by $H$ within *cspec* is as follows:
$\{(Z, z(f)) : com \models (Z, z(f))$ for each $\Psi$ interpretation $com$ such that $com \models \langle \Psi, \{Ax^r \cup H^r\}_{r \in AS} \rangle\}$.
Whenever $(Z, z(f)) \in Ax^{H \models_{\Psi^r}}$ we write $H \models_{cspec^r} (Z, z(f))$.

The following results depict the three main forms of inheritance imposed by a global specification on its local theories. Although interesting in themselves, they appear herein as lemmas for the soundness result presented in the next section. It is straightforward to prove them taking into account the tight binding between life cycles of a community interpretation structure (clauses (2) and (3) of definition 5.4). The third result deserves some attention because it might easily be overlooked: interactions are also inherited via specialization!

**Theorem 5.13** Let $cspec = \langle \Psi, \{Ax^r\}_{r \in AS} \rangle$ be a community specification with signature $\Psi = \langle S_{GS,\leq}, \{\Phi^r\}_{r \in AS} \rangle$. For each pair $r, r' \in GS$, if $r \leq r'$ then $\eta_{r',r} : \langle \Phi^{r'}, Ax^{r'} \rangle \to \langle \Phi^r, Ax^{\models_{\Psi^r}} \rangle$ is an aspect specialization morphism.

**Theorem 5.14** Let $cspec = \langle \Psi, \{Ax^r\}_{r \in AS} \rangle$ be a community specification with signature $\Psi = \langle S_{GS,\leq}, \{\Phi^r\}_{r \in AS} \rangle$. If $r = r_1 \times ... \times r_n$ with $r_1, ..., r_n \in GS$ then $\mu_i : \langle \Phi^{r_i}, Ax^{r_i} \rangle \to \langle \Phi^r, Ax^{\models_{\Psi^r}} \rangle$ is an aspect incorporation morphism for each $i = 1, .., n$.

**Theorem 5.15** Let $cspec = \langle \Psi, \{Ax^r\}_{r \in AS} \rangle$ be a community specification with signature $\Psi = \langle S_{GS,\leq}, \{\Phi^r\}_{r \in AS} \rangle$. If $r = r_1 \times ... \times r_n$ and $r' = r'_1 \times ... \times r'_n$ with $r_i \leq r'_i$ for $i = 1, ..., n$ then $\eta_{r'_1,r_1} \times ... \times \eta_{r'_n,r_n} : \langle \Phi^{r'}, Ax^{r'} \rangle \to \langle \Phi^r, Ax^{\models_{\Psi^r}} \rangle$ is an aspect specification morphism.

## 5.3 Global reasoning

The objective now is to introduce a deductive system for reasoning about a community taking into account inheritance and interaction. Note again the indexing on aspect sorts throughout.

**Definition 5.16** Let $cspec = \langle \Psi, Ax \rangle$ be a community specification with signature $\Psi = \langle S_{GS,\leq}, \{\Phi^r\}_{r \in AS} \rangle$ and $Ax = \{Ax^r\}_{r \in AS}$. The *theory generated* by *cspec* is the family $Ax^{\vdash_\Psi} = \{Ax^{\vdash_{\Psi^r}}\}_{r \in AS}$ such that $Ax^{\vdash_{\Psi^r}}$ is the least set of formulae of $L_{\Phi^r}$ satisfying the following requirements:

1. $(Z, z(f)) \in Ax^{\vdash_{\Psi^r}}$
   whenever $\vdash_{spec^r} (Z, z(f))$ where $spec^r = \langle \Phi^r, Ax^r \rangle$: *LocA*;
2. If $(Z, z(f)), (Z, z(f \Rightarrow g)) \in Ax^{\vdash_{\Psi^r}}$ then $(Z, z(g)) \in Ax^{\vdash_{\Psi^r}}$: *DetR*;
3. If $(Z, z(f)) \in Ax^{\vdash_{\Psi^r}}$ then $(Z, z(\forall x f)) \in Ax^{\vdash_{\Psi^r}}$: *GenR*;
4. If $(Z, z(f)) \in Ax^{\vdash_{\Psi^r}}$ then $(Z, z(\mathsf{G}\, f)) \in Ax^{\vdash_{\Psi^r}}$: $\mathsf{G}\, NecR$;
5. If $(Z, z(f)) \in Ax^{\vdash_{\Psi^r}}$ then $(Z, z(\mathsf{X}\, f)) \in Ax^{\vdash_{\Psi^r}}$: $\mathsf{X}\, NecR$;
6. $\eta_{r',r}(Z, z(f)) \in Ax^{\vdash_{\Psi^r}}$
   provided that $(Z, z(f)) \in Ax^{\vdash_{\Psi^{r'}}}$ whenever $r \leq r'$: *SpcR*;
7. $\mu_i(Z, z(f)) \in Ax^{\vdash_{\Psi^r}}$
   provided that $(Z, z(f)) \in Ax^{\vdash_{\Psi^{r_i}}}$ for some $i = 1, ..., n$ whenever $r = r_1 \times ... \times r_n$ with $r_1, ..., r_n \in GS$: *IncR*;

8. $\eta_{r'_1,r_1} \times ... \times \eta_{r'_n,r_n}(Z, z(f)) \in Ax^{\vdash_{\Psi}r}$
provided that $(Z, z(f)) \in Ax^{\vdash_{\Psi}r'}$ whenever $r = r_1 \times ... \times r_n$ and $r' = r'_1 \times ... \times r'_n$ with $r_i \leq r'_i$ for $i = 1, ..., n$: *IntR*;

9. $(Z, (r(z) \Rightarrow z(f))) \in Ax^{\vdash_{\Psi}r'}$
provided that $r \leq r'$ and $(Z_{\eta_{r',r}}, z(\eta_{r',r}(f))) \in Ax^{\vdash_{\Psi}r}$: *SRefR*;

10. $(Z, z(f)) \in Ax^{\vdash_{\Psi}r_0}$
provided that $r = r_0 \times ... \times r_n$, the set of free aspect sort variables in $z(f)$ is $\{z, x_1, ..., x_n\}$, the sort of $x_i$ is $r_i$ for $i = 1, ..., n$ and $(Z_{\mu_0}, z(g)) \in Ax^{\vdash_{\Psi}r}$ where $g$ is obtained from $\mu_0(f)$ by replacing $x_i$ by $\pi_i(z)$ for $i = 1, ..., n$: *IRefR*.

**Definition 5.17** Let $cspec = \langle \Psi, Ax \rangle$ be a community specification. Each element $(Z, z(f))$ of $Ax^{\vdash_{\Psi}r}$ for each $r \in AS$ is said to be a *r-theorem* of *cspec*, written $\vdash_{cspec^r} (Z, z(f))$.

The axiom schema *LocA* brings in all local theorems. Inference rule *SpcR* imposes that the $r'$-theorems are mapped onto $r$-theorems whenever $r \leq r'$. Inference rule *IncR* imposes that, for $i = 1, ..., n$, the $r_i$-theorems are mapped onto $r$-theorems whenever $r = r_1 \times ... \times r_n$ with $r_1, ..., r_n \in GS$. Inference rule *IntR* makes sure that interactions are also inherited. On the other hand, *SrefR* imposes that some $r$-theorems are reflected as $r'$-theorems for $r \leq r'$. Rule *IRefR* imposes a similar reflection for components of aggregations.

**Definition 5.18** Let $cspec = \langle \Psi, \{Ax^r\}_{r \in AS} \rangle$ be a community specification with signature $\Psi = \langle S_{GS,\leq}, \{\Phi^r\}_{r \in AS} \rangle$ and $H = \{H^r\}_{r \in AS}$ where each $H^r$ is a set of formulae of $L_{\Phi^r}$. The set of formulae *derived* from $H$ within *cspec* is the least set of formulae of $L_{\Phi^r}$, denoted $Ax^{H \vdash_{\Psi}r}$, satisfying the following requirements:

- $Ax^{\vdash_{\Psi}r} \subseteq Ax^{H \vdash_{\Psi}r}$;
- $H^r \subseteq Ax^{H \vdash_{\Psi}r}$;
- If $(Z, z(f)), (Z, z(f \Rightarrow g)) \in Ax^{H \vdash_{\Psi}r}$ then $(Z, z(g)) \in Ax^{H \vdash_{\Psi}r}$: *DetR*.

We write $H \vdash_{cspec^r} (Z, z(f))$ whenever $(Z, z(f)) \in Ax^{H \vdash_{\Psi}r}$.

The soundness of the proposed global deductive system should not be too difficult to establish since the specific inference rules mirror the tight relationships imposed in the interpretation structures between the different life cycles:

**Theorem 5.19** (Soundness Theorem)
Let $cspec = \langle \Psi, Ax \rangle$ be a community specification, $H = \{H^r\}_{r \in AS}$ where each $H^r$ is a set of formulae of $L_{\Phi^r}$ and $(Z, z(f)) \in L_{\Phi^r}$. If $H \vdash_{cspec^r} (Z, z(f))$ then $H \vDash_{cspec^r} (Z, z(f))$.

Proof: The result is obtained by induction following the inductive definition of the set $Ax^{H \vdash_{\Psi}r}$. The soundness of the inheritance rules *SpcR*, *IncR* and *IntR* is an immediate consequence of the three results stated at the end of the previous section. The reflection rules *SRefR* and *IRefR* deserve some attention. But since the former is much easier to establish than the latter we outline herein only the proof of the soundness of *IRefR*. We have to prove that if $(Z_{\mu_0}, z(g)) \in Ax^{\vdash_{\Psi}r}$ then $(Z, z(f)) \in Ax^{\vdash_{\Psi}r_0}$ (provided that $r = r_0 \times ... \times r_n$, the set of free aspect sort variables in $z(f)$ is $\{z, x_1, ..., x_n\}$, the sort of $x_i$ is $r_i$ for $i = 1, ..., n$, and $g$ is obtained from $\mu_0(f)$ by replacing $x_i$ by $\pi_i(z)$ for $i = 1, ...n$). For the sake of simplicity we consider $f$ to be an atomic formula $p(z, x_1, ...., x_n)$: there is no trouble in retaining the result in the general case. Therefore, we need to verify that if $com \vDash z.\mu_0(p)(\pi_0(z), \pi_1(z), ..., \pi_n(z))$ for every structure $com$ satisfying $cspec$ then $com \vDash z.p(z, x_1, ..., x_n)$ for every structure $com$ satisfying $cspec$. Actually, we verify that if $com \vDash z.\mu_0(p)(\pi_0(z), \pi_1(z), ..., \pi_n(z))$ then $com \vDash z.p(z, x_1, ..., x_n)$. Indeed, from the former we successively conclude: $com \vDash_{asg,k} z.\mu_0(p)(\pi_0, \pi_1(z), ..., \pi_n(z))$ for every assignment $asg$ and position $k$; $asp^r \vDash_{asg^r,k} z.\mu_0(p)(\pi_0(z), \pi_1(z), ..., \pi_n(z))$ for every $asg$ and $k$; $asg^r(z).\mu_0(p)(asg^r(z)_0, asg^r(z)_1, ..., asg^r(z)_n) \in \lambda^r_k$ for every $asg$ and $k$; $\mu_0(p)(asg^r(z)_0, asg^r(z)_1, ..., asg^r(z)_n) \in \lambda^r_{asg^r(z),k}$ for every $asg$ and $k$; $\mu_0(p)(a_0, a_1, ..., a_n) \in \lambda^r_{(a_0,a_1,...,a_n),k}$ for every $a_0, a_1, ..., a_n$ and $k$; $p(a_0, a_1, ..., a_n) \in \lambda^{r_0}_{a_0,k}$ for every $a_0, a_1, ..., a_n$ and $k$ (using clause (3) in the definition of community interpretation structure); $p(asg^{r_0}(z)_0, asg^{r_0}(x_1), ..., asg^r(x_n)) \in \lambda^{r_0}_{asg^{r_0}(z),k}$ for every $asg$ and $k$; $asp^{r_0} \vDash_{asg^{r_0},k} z.p(z, x_1, ..., x_n)$ for every $asg$ and $k$; $com \vDash z.p(z, x_1, ..., x_n)$.

Again, the following metatheorems are useful:

**Theorem 5.20** (Metatheorems)
Let $cspec = \langle \Psi, Ax\rangle$ be a community specification and $(Z, z(f)), (Z', z'(f')) \in L_{\Phi^r}$. Then:

- If $(Z, z(f)) \vdash_{cspec^r} (Z', z'(f'))$ then $\vdash_{cspec^r} ((Z, z(f)) \Rightarrow (Z', z'(f')))$: *DED*;
- If $(Z, z(f)) \vdash_{cspec^r} (Z', z'(f'))$ and $(Z, z(f)) \vdash_{cspec^r} (Z', z'(\neg f'))$ then $\vdash_{cspec^r} (Z, z(\neg f))$: *ABS*.

**Example 5.21**

$$\vdash^{book\times user}_{cspec_{library}} (z.occistaken(\pi_2(z)) \Rightarrow (\mathsf{F}\, z.occisreturned(\pi_2(z))))$$

Derivation:

| | | |
|---|---|---|
| 0. | $z.occistaken(\pi_2(z))$ | $Hyp$ |
| 1. | $(z.occistaken(\pi_2(z)) \Rightarrow z.occtakes(\pi_1(z)))$ | $LocA^{book\times user}$ |
| 2. | $z.occtakes(\pi_1(z))$ | $DetR : 0, 1$ |
| 3. | $(z.occtakes(\pi_1(z))\Rightarrow$ $(z.borrowedbook(\pi_1(z), true)\ \mathsf{U}\ z.occreturns(\pi_1(z))))$ | $IncR^{user}$ |
| 4. | $(z.borrowedbook(\pi_1(z), true)\ \mathsf{U}\ z.occreturns(\pi_1(z)))$ | $DetR : 2, 3$ |
| 5. | $((z.borrowedbook(\pi_1(z), true)\ \mathsf{U}\ z.occreturns(\pi_1(z)))\Rightarrow$ $(\mathsf{F}\, z.occreturns(\pi_1(z))))$ | $UFA$ |
| 6. | $(\mathsf{F}\, z.occreturns(\pi_1(z)))$ | $DetR : 4, 5$ |
| 7. | $(\mathsf{G}(z.occreturns(\pi_1(z)) \Rightarrow z.occisreturned(\pi_2(z))))$ | $LocA^{book\times user}, GNecR$ |
| 8. | $((\mathsf{F}(z.occreturns(\pi_1(z))) \Rightarrow (\mathsf{F}\, z.occisreturned(\pi_2(z))))$ | $TLT, DetR : 7$ |
| 9. | $(\mathsf{F}\, z.occisreturned(\pi_2(z)))$ | $DetR : 6, 8$ |
| 10. | $(z.occistaken(\pi_2(z)) \Rightarrow (\mathsf{F}\, z.occisreturned(\pi_2(z))))$ | $DED : 0, 9$ |

**Example 5.22**

$$\vdash^{book}_{cspec_{library}} (b.occistaken(u) \Rightarrow (\mathsf{F}\, b.occisreturned(u)))$$

Derivation:

| | | |
|---|---|---|
| 0. | $(z.occistaken(u) \Rightarrow (\mathsf{F}\, z.occisreturned(u)))$ | $IRefR : example$ 5.21 |
| 1. | $(b.occistaken(u) \Rightarrow (\mathsf{F}\, b.occisreturned(u)))$ | $FOT, Det : 0$ |

**Example 5.23** Assuming that $resbook \leq_{book} book$,

$$\vdash^{book}_{cspec_{library}} (resbook(b) \Rightarrow (\mathsf{X}(\mathsf{F}\, b.available(true))))$$

Derivation:

| | | |
|---|---|---|
| 0. | $(resbook(r) \Rightarrow (\mathsf{X}(\mathsf{F}\, r.available(true))))$ | $SRefR : example$ 3.25 |
| 1. | $(resbook(b) \Rightarrow (\mathsf{X}(\mathsf{F}\, b.available(true))))$ | $FOT, Det : 0$ |

These examples illustrate the main features of the proposed logic for reasoning globally about a community of object aspects: namely, importing local theorems, inheritance into specializations and aggregations, as well as reflection.

# 6 Concluding remarks

An object aspect specification logic was presented that provides the means for reasoning about communities of aspects, dealing with their aggregation without and with interaction plus specialization. The semantics is a straightforward extension of the traditional Kripke semantics for a many-sorted, first-order, linear temporal predicate logic. The extension includes:

1. the treatment of the non-rigid attributes as predicates (with the expected functionality constraint);
2. a rather weak constraint on sequentiality of components (mutual exclusion of similar actions) allowing simultaneous actions in aggregations;
3. a frame constraint (observations do not change attributes);
4. incorporation of the theorems about the underlying data structure;
5. inheritance of theorems into specializations;
6. inheritance of theorems into aggregations;
7. reflection of theorems from specializations;
8. reflection of theorems from aggregations.

The proposed reflection rules show that both specialization and aggregation induce in general non-conservative extensions of the original aspects (cf Examples 5.22 and 5.23). This is not unexpected and we consider it an essential feature of the proposed logic.

The inheritance rules clearly show that we have here a rather simple approach to specialization (and aggregation) of a purely monotonic nature: specialization (and aggregation) may only add new theorems. We are aware that such a monotonic approach may not be sufficient. For instance, specialization with overriding is non-monotonic. This question should be addressed in subsequent work, maybe along the lines proposed for semantic domains in [CSS94] using the notion of partial morphism. However, it does not seem trivial to lift such a notion to the level of theories. Work in this direction is already presented in [BLR93] based upon an earlier version of the logic presented herein.

Another limitation of the proposed logic is the impossibility of having consistent specializations with side-effects (that is, where a new action affects the value of an attribute of the original aspect). Indeed, the logic as it stands imposes a "no side effects constraint" (cf the tight relationship between the life-cycles of a specialization and its original aspect). The same applies to aggregations but in this case that is precisely what we want: in an aggregation we should not expect that an action of a component may affect the value of an attribute of another component. Providing a mechanism for supporting side effects in specializations seems to require a minor revision of the proposed logic in the direction of treating in a rather different way observations and actions (in the global semantics and in the corresponding rules of the deductive system).

The question of fairness was completely disregarded in this paper. Indeed, we assume that the task of imposing fairness or justice requirements is left to the specifier (to the point of inconsistency if by mistake too much is required). Further work is needed on this topic, adapting to the proposed framework the ideas already present in the literature (see for instance [Fra86, Jon90, MP92]). Some interesting results concerning fair aggregation are presented in [SS93], where an institution of objects is established for any given temporal logic of specification. In the institutional front see also [SCS94, FCSM93]. To this end it seems useful to bring in the distinction between action enabling and action occurrence.

The problem of the (relative) completeness of the axiomatic system we propose was also ignored in the paper. Indeed, given that we accept as theorems all that is known about data types and since the relative completeness of the temporal and first-order fragments was already established in the literature, what remains at stake is the completeness of the object aspect and community specific axioms and rules. This is an interesting topic for further research, maybe by coding first the entire logic into a pure many-sorted, first-order logic (see [Man93]).

Notwithstanding these limitations, the proposed logic does provide a useful setting for specifying and reasoning about object classes, their aggregations and specializations. We have been experimenting with practical examples within the context of a large-scale application and we are now working towards a more machine-oriented deductive system with automatic (or at least computer-aided) theorem proving in mind (see [GSGA93]). A prototype is running for a significant fragment of the proposed logic. Concerning reasoning about numbers the prototype uses the simplex algorithm (following [NO79]). The proposed logic has been shown to be useful for providing a (proof-theoretic) semantics of object-oriented specification languages: see for instance [Jun93].

# 7 Appendix: Freely generated cartesian categories

We shortly describe the functor $Fcc : PO \rightarrow Cat$ that maps each partial order into its canonical cartesian extension. Let $P = \langle |P|, \leq \rangle$ be a partial order. Then $Fcc(P)$ is the cartesian category such that:

1. $\langle |Fcc(P)|, \times, 1 \rangle$ is the monoid freely generated from $|P|$;
2. the set $Mor_{Fcc(P)}$ is the quocient set induced on the set inductively defined as follows:
   - $(\iota : s \rightarrow s') \in Mor_{Fcc(P)}$ provided that $s \leq s'$;
   - $(\iota : s \rightarrow s) \in Mor_{Fcc(P)}$ provided that $s \in (|Fcc(P)| \setminus |P|)$;
   - $(\tau : s \rightarrow 1) \in Mor_{Fcc(P)}$ provided that $s \in |Fcc(P)|$;
   - $(\pi_1 : s' \times s'' \rightarrow s') \in Mor_{Fcc(P)}$ provided that $s', s'' \in |Fcc(P)|$;
   - $(\pi_2 : s' \times s'' \rightarrow s'') \in Mor_{Fcc(P)}$ provided that $s', s'' \in |Fcc(P)|$;
   - $((\nu : s \rightarrow s'); (\nu' : s' \rightarrow s'') : s \rightarrow s'') \in Mor_{Fcc(P)}$
     provided that $\nu : s \rightarrow s', \nu' : s' \rightarrow s'' \in Mor_{Fcc(P)}$;

- $((\nu' : s \to s') \times (\nu'' : s \to s'') : s \to s' \times s'') \in Mor_{Fcc(P)}$ provided that $\nu' : s \to s', \nu'' : s \to s'' \in Mor_{Fcc(P)}$;

by the least equivalence relation induced by the equations:

- $((\iota : s \to s); (\nu : s \to s') : s \to s') = (\nu : s \to s')$;
- $((\nu : s \to s'); (\iota : s' \to s') : s \to s') = (\nu : s \to s')$;
- $((\iota : s \to s'); (\iota' : s' \to s'') : s \to s'') = (\iota : s \to s'')$;
- $(((\nu : s \to s'); (\nu' : s' \to s'') : s \to s''); (\nu'' : s'' \to s''') : s \to s''')$ $= ((\nu : s \to s'); ((\nu' : s' \to s''); ((\nu'' : s'' \to s''') : s \to s''') : s \to s''')$;
- $(((\nu' : s \to s') \times (\nu'' : s \to s'') : s \to s' \times s''); (\pi_1 : s' \times s'' \to s') : s \to s') = \nu' : s \to s')$;
- $(((\nu' : s \to s') \times (\nu'' : s \to s'') : s \to s' \times s''); (\pi_2 : s' \times s'' \to s'') : s \to s'') = \nu' : s \to s'')$;
- $(((\nu : s \to s' \times s''); (\pi_1 : s' \times s'' \to s') : s \to s') \times ((\nu : s \to s' \times s''); (\pi_2 : s' \times s'' \to s'') : s \to s'') : s \to s' \times s'') = \nu : s \to s' \times s'')$;
- $(\tau : s \to 1) = (\nu : s \to 1)$;

3. the identity in each $s \in |Fcc(P)|$ is $(\iota : s \to s)$;
4. the composition of $(\nu : s \to s')$ and $(\nu' : s' \to s'')$ is $((\nu : s \to s'); (\nu' : s' \to s'') : s \to s'')$.

## Acknowledgments

This work was partially supported by the JNICT Projects PMCT/C/TIT/177-178/90 (OBCALC and FAC3), the ESPRIT Basic Research Actions 6071-6112 (IS-CORE and COMPASS), and, last but not least, by the ESDI Project OBLOG. The authors are grateful to their many colleagues in these five projects for many rewarding interactions, specially to José Fiadeiro, Tom Maibaum and Udo Lipeck. The authors are also grateful to the anonymous referees for many suggestions towards improving the readability of the paper for a wider audience.

## References

[AHS90] J. Adámek, H. Herrlich, and G. Strecker. *Abstract and Concrete Categories*. John-Wiley, 1990.

[BHPS93] C. Beierle, U. Hedtstuck, U. Pletat, and J. Siekmann. An order-sorted predicate logic with closely coupled taxonomic information. In K. Meinke and J. Tucker, editors, *Many-sorted Logic and its Applications*, pages 179–211. John Wiley, 1993.

[BL91] S. Brass and U. Lipeck. Semantics of inheritance in logical object specifications. In *Proceedings of the 2nd International Conference on Deductive and Object-oriented Databases*, 1991.

[BLR93] S. Brass, U. Lipeck, and P. Resende. Specification of object behaviour with defaults. In U. Lipeck and G. Koschorreck, editors, *Proceedings of the International Workshop on Information Systems – Correctness and Reusability*, pages 155–177. University of Hannover, 1993.

[Boo86] G. Booch. Object oriented development. *IEEE Transactions on Software Engineering*, 12(2), 1986.

[Bro90] F. Brooks. No silver bullet. In T. DeMarco and T. Lister, editors, *Software state-of-the-art*, pages 14–29. Dorset House Publishing, 1990.

[Cox86] B. Cox. *Object Oriented Programming: an Evolutionary Approach.* Addison-Wesley, 1986.

[Cox90] B. Cox. There is a silver bullet. *Byte*, pages 209–218, 1990.

[CS91] J. Carmo and A. Sernadas. Formal techniques for systems specification and verification. *Information Systems*, 16:245–272, 1991.

[CSS94] J. Costa, A. Sernadas, and C. Sernadas. Object inheritance beyond subtyping. *Acta Informatica*, 31:5–26, 1994.

[DMN67] O.-J. Dahl, B. Myhrhaug, and K. Nygaard. Simula 67: Common base language. Research report, Norwegian Computer Centre, 1967.

[EM85] H. Ehrig and B. Mahr. *Fundamentals of Algebraic Specification I.* Springer-Verlag, 1985.

[Eme90] E. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science B: Formal Models and Semantics*, pages 995–1072. Elsevier, 1990.

[End72] H. Enderton. *A Mathematical Introduction to Logic.* Academic Press, 1972.

[ES91] H.-D. Ehrich and A. Sernadas. Object concepts and constructions. In G. Saake and A. Sernadas, editors, *Proceedings of the IS-CORE Workshop 91*, pages 1–24. Technical University of Braunschweig, 1991.

[FCSM93] J. Fiadeiro, J. Costa, A. Sernadas, and T. Maibaum. Process semantics of temporal logic specification. In M. Bidoit and C. Choppy, editors, *Recent Trends in Data Type Specification: 8th Workshop on Specification of Abstract Data Types – Selected papers*, pages 236–253. Springer-Verlag, 1993.

[FM92] J. Fiadeiro and T. Maibaum. Temporal theories as modularization units for concurrent system specification. *Formal Aspects of Computing*, 4:239–272, 1992.

[Fra86] N. Francez. *Fairness.* Springer-Verlag, 1986.

[FS88] J. Fiadeiro and A. Sernadas. Specification and verification of database dynamics. *Acta Informatica*, 25:625–661, 1988.

[FSMS91] J. Fiadeiro, C. Sernadas, T. Maibaum, and G. Saake. Proof-theoretic semantics of object-oriented specification constructs. In R. Meersman, W. Kent, and S. Khosla, editors, *Object-oriented Databases: Analysis, Design and Construction*, pages 243–284. North-Holland, 1991.

[GB92] J. Goguen and R. Burstall. Institutions: Abstract model theory for specification and programming. *Journal of the ACM*, 39(1):95–146, 1992.

[Gol87] R. Goldblatt. *Logics of Time and Computation.* CSLI, 1987.

[GR83] A. Goldberg and D. Robson. *SmallTalk80: The Language and its Implementation.* Addison-Wesley, 1983.

[GSGA93] P. Gouveia, C. Sernadas, J. Gomes, and J. Apolinário. Tableaux for reasoning about objects. In *Theorem Proving with Analytic Tableaux and Related Methods*, pages 113–125, 1993. Max Plank Institut fur Informatik.

[Jon90] B. Jonsson. Modular verification of asynchronous networks. In *Proceedings of Symposium on Mathematical Foundations of Computer Science*, pages 347–354. Springer-Verlag, 1990.

[JSHS91] R. Jungclaus, G. Saake, T. Hartmann, and C. Sernadas. *Object-oriented specification of information systems: The TROLL language*. Technical University of Braunschweig, 1991.

[Jun93] R. Jungclaus. *Modeling of Dynamic Object Systems: A Logic-based Approach*. Vieweg, 1993.

[LS91] R. Li and A. Sernadas. Reasoning about objects using tableau method. *Journal of Logic and Computation*, 1:575–611, 1991.

[Man93] M. Manzano. Introduction to many-sorted logic. In K. Meinke and J. Tucker, editors, *Many-sorted Logic and its Applications*, pages 3–86. John Wiley, 1993.

[McA76] R. McArthur. *Tense Logic*. Reidel, 1976.

[Mey88] B. Meyer. *Object Oriented Software Construction*. Prentice Hall, 1988.

[MP92] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems*. Springer-Verlag, 1992.

[MT93] K. Meinke and J. Tucker. *Many-sorted Logic and its Applications*. John Wiley, 1993.

[Nie89] O. Nierstrasz. Process semantics of temporal logic specification. In W. Kim and F. Lochovski, editors, *Object oriented concepts*, pages 3–22. ACM Press, 1989.

[NO79] G. Nelson and D. Oppen. Simplification by cooperating decision procedures. *ACM Transactions on Programming Languages and Systems*, 1:245–257, 1979.

[Pnu77] A. Pnueli. The temporal logic of programs. In *Proceedings of the 18th IEEE Symposium on Foundations of Computer Science*, pages 45–57, 1977.

[SCS94] A. Sernadas, J. Costa, and C. Sernadas. An institution of object behaviour. In H. Ehrig and F. Orejas, editors, *Recent Trends in Data Type Specification: 9th Workshop on Specification of Abstract Data Types – Selected papers*, pages 337–350. Springer-Verlag, 1994.

[SE91] A. Sernadas and H.-D. Ehrich. What is an object after all? In R. Meersman, W. Kent, and S. Khosla, editors, *Object-oriented Databases: Analysis, Design and Construction*, pages 39–69. North-Holland, 1991.

[Ser80] A. Sernadas. Temporal aspects of logical procedure definition. *Information Systems*, 5:167–187, 1980.

[SFSE89a] A. Sernadas, J. Fiadeiro, C. Sernadas, and H.-D. Ehrich. Abstract object types: A temporal perspective. In B. Banieqbal, H. Barringer, and A. Pnueli, editors, *Temporal Logic in Specification*, pages 324–350. Springer-Verlag, 1989.

[SFSE89b] A. Sernadas, J. Fiadeiro, C. Sernadas, and H.-D. Ehrich. The basic building blocks of information systems. In E. Falkenberg and P. Lindgreen, editors, *Information Systems Concepts: An In-depth Analysis*, pages 225–246. North-Holland, 1989.

[SGL92] C. Sernadas, P. Gouveia, and A. Lopes. Gentzen-type system for verification in conceptual modeling. In J. Goers, A. Hever, and G. Saake, editors, *Foundations of Models and Languages*, pages 215–228. 1992.

[SRGS91] C. Sernadas, P. Resende, P. Gouveia, and A. Sernadas. In-the-large object-oriented design of information systems. In F. van Assche, B. Moulin, and C. Rolland, editors, *The Object-oriented Approach in Information Systems*, pages 209–232. North-Holland, 1991.

[SS93] A. Sernadas and C. Sernadas. Denotational semantics of object specification within an arbitrary temporal logic institution. Research report, Section of Computer Science, Department of Mathematics, Instituto Superior Técnico, 1096 Lisboa, Portugal, 1993. Presented at IS-CORE Workshop 93 - Submitted for publication.

[Str86] B. Stroustrup. *The C++ Programming Language.* Addison-Wesley, 1986.

[Sza78] M. Szabo. *Algebra of Proofs.* North-Holland, 1978.

[TZ88] J. Tucker and J. Zucker. *Program Correctness over Abstract Data Types with Error-state Semantics.* North-Holland, 1988.