Logical Foundations of CafeOBJ

Răzvan Diaconescu [1] and Kokichi Futatsugi

Japan Advanced Institute of Science and Technology [2]

Abstract

This paper surveys the logical and mathematical foundations of CafeOBJ, which is a successor of the famous algebraic specification language OBJ but adds to it several new primitive paradigms such as behavioural concurrent specification and rewriting logic.

We first give a concise overview of CafeOBJ. Then we focus on the actual logical foundations of the language at two different levels: basic specification and structured specification, including also the definition of the CafeOBJ institution. We survey some novel or more classical theoretical concepts supporting the logical foundations of CafeOBJ, pointing out the main results but without giving proofs and without discussing all mathematical details. Novel theoretical concepts include the *coherent hidden algebra* formalism and its combination with rewriting logic, and *Grothendieck* (or *fibred*) *institutions*. However for proofs and for some of the mathematical details not discussed here we give pointers to relevant publications.

The logical foundations of CafeOBJ are structured by the concept of *institution*. Moreover, the design of CafeOBJ emerged from its logical foundations, and institution concepts played a crucial rôle in structuring the language design.

Key words: algebraic specification, CafeOBJ, institutions, behavioural specification

1 Introduction

CafeOBJ is an *executable* industrial strength algebraic specification language which is a modern successor of OBJ and incorporates several new algebraic specification paradigms. Its definition is given in [13]. CafeOBJ is intended to be mainly used for system specification, formal verification of specifications, rapid prototyping, or even programming. We give below a brief overview of its most important features.

[1] On leave from the Institute of Mathematics of the Romanian Academy.

[2] 1-1 Asahidai, Tatsunokuchi, Ishikawa 923-1292, JAPAN

Equational Specification and Programming.

Equational specification and programming is inherited from OBJ [29, 19] and constitutes the basis of the language, the other features being somehow built on top of it. As with OBJ, CafeOBJ is *executable* (by term rewriting), which gives an elegant declarative way of functional programming, often referred as *algebraic programming*. [3] As with OBJ, CafeOBJ also permits equational specification modulo several equational theories such as associativity, commutativity, identity, idempotence, and combinations between all these. This feature is reflected at the execution level by term rewriting *modulo* such equational theories.

Behavioural Specification.

Behavioural specification [23, 24, 14, 31] provides a novel generalisation of ordinary algebraic specification. Behavioural specification characterises how objects (and systems) *behave*, not how they are implemented. This new form of abstraction can be very powerful in the specification and verification of software systems since it naturally embeds other useful paradigms such as concurrency, object-orientation, constraints, nondeterminism, etc. (see [24] for details). Behavioural abstraction is achieved by using specification with hidden sorts and a behavioural concept of satisfaction based on the idea of indistinguishability of states that are observationally the same, which also generalises process algebra and transition systems (see [24]). CafeOBJ behavioural specification paradigm is based on *coherent hidden algebra* (abbreviated 'CHA') of [14], which is both a simplification and extension of classical hidden algebra of [24] in several directions, most notably by allowing operations with multiple hidden sorts in the arity. Coherent hidden algebra comes very close to the "observational logic" of Bidoit and Hennicker [31].

CafeOBJ directly supports behavioural specification and its proof theory through special language constructs, such as

- hidden sorts (for states of systems),
- behavioural operations (for direct "actions" and "observations" on states of systems),
- behavioural coherence declarations for (non-behavioural) operations (which may be either derived (indirect) "observations" or "constructors" on states of systems), and
- behavioural axioms (stating behavioural satisfaction).

The advanced coinduction proof method receives support in CafeOBJ via a default (candidate) coinduction relation (denoted =*=). In CafeOBJ, coinduction can be used either in the classical hidden algebra sense [24] for proving behavioural

[3] Although this paradigm may be used as programming, from the applications point of view, this aspect is secondary to its specification side.

equivalence of states of objects, or for proving behavioural transitions (which appear when applying behavioural abstraction to rewriting logic). However, until the time this paper was written, the latter has not been yet explored sufficiently, especially practically.

Besides language constructs, CafeOBJ supports behavioural specification and verification by several methodologies.[4] CafeOBJ currently highlights a methodology for concurrent object composition which features high reusability not only of specification code but also of verifications [13, 33]. Behavioural specification in CafeOBJ may also be effectively used as an object-oriented (state-oriented) alternative for classical data-oriented specifications. Experiments seem to indicate that an object-oriented style of specification even of basic data types (such as sets, lists, etc.) may lead to higher simplicity of code and drastic simplification of verification process [13].

Behavioural specification is reflected at the execution level by the concept of *behavioural rewriting* [13, 14] which refines ordinary rewriting with a condition ensuring the correctness of the use of behavioural equations in proving strict equalities.

Rewriting Logic Specification.

Rewriting logic specification in CafeOBJ is based on a simplified version of Meseguer's *rewriting logic* (abbreviated as 'RWL') [36] specification framework for concurrent systems which gives a non-trivial extension of traditional algebraic specification towards concurrency. RWL incorporates many different models of concurrency in a natural, simple, and elegant way, thus giving CafeOBJ a wide range of applications. Unlike Maude [3], the current CafeOBJ design does not fully support *labelled* RWL which permits full reasoning about multiple transitions between states (or system configurations), but provides proof support for reasoning about the *existence* of transitions between states (or configurations) of concurrent systems via a built-in predicate (denoted ==>) with dynamic definition encoding into equational logic both the proof theory of RWL and the user defined transitions (rules). At the level of the semantics, this amounts to the fact that the CafeOBJ RWL models are preorders rather than categories.

From a methodological perspective, CafeOBJ develops the use of RWL transitions for specifying and verifying the properties of *declarative encoding of algorithms* (see [13]) as well as for specifying and verifying transition systems.

[4] This is still an open research topic, the current methodologies may be developed further and new methodologies may be added in the future.

Module System.

The principles of the CafeOBJ module system are inherited from OBJ which builds on ideas first realized in the language Clear [1], most notably institutions [21, 17]. CafeOBJ module system features

- several kinds of imports,
- sharing for multiple imports,
- parameterised programming allowing
  - multiple parameters,
  - views for parameter instantiation,
  - integration of CafeOBJ specifications with executable code in a lower level language
- module expressions.

However, the concrete design of the language revises the OBJ view on importation modes and parameters [13].

Type System and Partiality.

CafeOBJ has a type system that allows subtypes based on *order sorted algebra* (abbreviated 'OSA') [27, 22]. This provides a mathematically rigorous form of runtime type checking and error handling, giving CafeOBJ a syntactic flexibility comparable to that of untyped languages, while preserving all the advantages of strong typing.

At this moment the concrete order sortedness formalism is still open at least at the level of the language definition. CafeOBJ does not directly do partial operations but rather handles them by using error sorts and a sort membership predicate in the style of *membership equational logic* (abbreviated 'MEL') [37]. The semantics of specifications with partial operations is given by MEL.

Logical semantics.

CafeOBJ is a declarative language with firm mathematical and logical foundations in the same way as other OBJ-family languages (OBJ, Eqlog [25, 5], FOOPS [26], Maude [36]) are. The mathematical semantics of CafeOBJ is based on state-of-the-art algebraic specification concepts and results, and is strongly based on category theory and the theory of institutions [21, 12, 10, 17]. The following are the principles governing the logical and mathematical foundations of CafeOBJ:

P1. there is an underlying logic [5] in which all basic constructs and features of

[5] Here "logic" should be understood in the modern relativistic sense of "institution" which

the language can be rigorously explained.

P2. provide an integrated, cohesive, and unitary approach to the semantics of specification in-the-small and in-the-large.

P3. develop all ingredients (concepts, results, etc.) at the highest appropriate level of abstraction.

CafeOBJ is a multi-paradigm language. Each of the main paradigms implemented in CafeOBJ is rigorously based on some underlying logic; the paradigms resulting from various combinations are based on the combination of logics. The structure of these logics is shown by the following *CafeOBJ cube*, where the full arrows mean embedding between the logics, which correspond to institution embeddings (i.e., a strong form of institution morphisms of [21, 17]) (the orientation of arrows goes from "more complex" to "less complex" logics in the style of the original definition of institution homomorphism [21]).

The mathematical structure represented by this cube is that of an *indexed institution* [12]. The CafeOBJ institution is a *Grothendieck* (or *fibred*) *institution* [12] obtained by applying a Grothendieck construction to this cube (i.e., the indexed institution). The dotted arrows represent the institution homomorphism [21]. Note that by employing other logical-based paradigms the CafeOBJ cube may be thought as a hyper-cube (see [13] for details).

1.1 Summary of the paper

The first part of this paper is dedicated to the foundations of basic specifications. The main topic of this part is the definition of HOSRWL, the hidden order sorted rewriting logic institution, which embeds all other institutions of the CafeOBJ cube. In this way, the HOSRWL institution contains the mathematical foundations for all basic specification CafeOBJ constructs.

provides a mathematical definition for a logic (see [21]) rather than in the more classical sense.

The second part of the paper presents the novel concept of Grothendieck institution (developed in [12]) which constructs the CafeOBJ institution from the CafeOBJ cube.

The last section contains the definitions of the main mathematical concepts for structuring specification in CafeOBJ.

The main concepts of the logical foundations of CafeOBJ are illustrated with several examples, including CafeOBJ code. We assume familiarity with CafeOBJ including its syntax and semantics (see [13] or several papers such as [15]).

Terminology and Notations

This work assumes some familiarity with basic general algebra (in its many-sorted and order-sorted form) and category theory. Relevant background in general algebra can be found in [20, 28, 38] for the many-sorted version, and in [27, 22] for the order-sorted version. For category theory we generally use the same notations and terminology as Mac Lane [34], except that composition is denoted by ";" and written in the diagrammatic order. The application of functions (functors) to arguments may be written either normally using parentheses, or else in diagrammatic order without parentheses, or, more rarely, by using sub-scripts or super-scripts. The category of sets is denoted as $\mathbb{S}et$, and the category of categories [6] as $\mathbb{C}at$. The opposite of a category $\mathbb{C}$ is denoted by $\mathbb{C}^{\mathrm{op}}$. The class of objects of a category $\mathbb{C}$ is denoted by $|\mathbb{C}|$; also the set of arrows in $\mathbb{C}$ having the object a as source and the object b as target is denoted as $\mathbb{C}(a,b)$. A *preorder* is a small category with *at most* one arrow between each two objects. A *preorder functor* is just a functor between preorders. The category of preorders is denoted by $\mathbb{P}re$.

Indexed categories [39] play an important rôle in this work. [40] constitutes a good reference for indexed categories and their applications to algebraic specification. An *indexed category* [40] is a functor $B\colon I^{\mathrm{op}} \to \mathbb{C}at$; sometimes we denote $B(i)$ as B_i (or B^i) for an index $i \in |I|$ and $B(u)$ as B^u for an index morphism $u \in I$. The following 'flattening' construction providing the canonical fibration associated to an indexed category is known under the name of the *Grothendieck construction*, and plays an important rôle in mathematics and in particular in this paper. Given an indexed category $B\colon I^{\mathrm{op}} \to \mathbb{C}at$, let $B^\sharp$ be the *Grothendieck category* having $\langle i, \Sigma\rangle$, with $i \in |I|$ and $\Sigma \in |B_i|$, as objects and $\langle u, \varphi\rangle\colon \langle i, \Sigma\rangle \to \langle i', \Sigma'\rangle$, with $u \in I(i,i')$ and $\varphi\colon \Sigma \to \Sigma' B^u$, as arrows. The composition of arrows in $B^\sharp$ is defined by $\langle u, \varphi\rangle; \langle u', \varphi'\rangle = \langle u;u', \varphi;(\varphi' B^u)\rangle$.

[6] We steer clear of any foundational problem related to the "category of all categories"; several solutions can be found in the literature, see, for example [34].

2 Foundations of Basic Specifications

At the level of the basic specifications, semantics of CafeOBJ is concerned with the semantics of collections of specification statements. CafeOBJ modules can be flattened to such *basic specifications* by an obvious induction process on the module composition structure. In CafeOBJ we can have several kinds of specifications, the basic kinds corresponding to the basic CafeOBJ specification/programming paradigms:

- equational specifications,
- rewriting specifications,
- behavioural specifications, and
- behavioural rewriting specifications.

The membership of a basic specification to a certain class is determined by the CafeOBJ convention that each basic specification should be regarded as implementing the simplest possible combination of paradigms resulting from its syntactic content.

2.1 Loose and Tight Denotation

The key concept of specification in-the-small is the *satisfaction relation* between the models and the sentences of a given specification, which is also the key notion of the abstract concept of institution. Each kind of specification has its own concept of satisfaction, and Section 2.2 surveys them briefly.

Each class of basic specifications has an underlying logic in the CafeOBJ cube. Specifications can be regarded as finite sets of sentences in the underlying logic. This enables us to formulate the principle of semantics of CafeOBJ specification in-the-small:

> **(S)** Each basic specification determines a theory in the corresponding institution. The **denotation** $[\![SP]\!]$ of a basic specification SP is the class of models $\mathrm{MOD}(T^{SP})$ of its corresponding theory T^{SP} if **loose**, and it is the initial model $0_{T^{SP}}$ of the theory, if **tight**.

A basic specification can have either loose or initial denotation, and this can be directly specified by the user. CafeOBJ does not directly implement final semantics, however final models play an important rôle for the loose semantics of behavioural specifications (see [14, 9]).

Initial model semantics applies only to non-behavioural specification, and is supported by the following result:

Theorem 1 *Let T be a theory in either MSA, OSA, RWL, or OSRWL. Then the initial model 0_T exists.*

This very important result appears in various variants and can be regarded as a classic of algebraic specification theory. The reader may wish to consult [28] for MSA, [27, 22] for OSA, [36] for RWL, and although, up to our knowledge, the result has not yet been published, it is also valid for OSRWL.

Because of the importance of the construction of the initial model we briefly recall it here. Let Σ be the signature of the theory consisting of a set S of sorts (which is a partial order in the order-sorted case) and a ranked (by S^*) set of operation symbols (possibly overloaded). The S-sorted set T_Σ of Σ-terms is the least S-sorted set closed under:

- each constant is a Σ-term (that is, $\Sigma_{[],s} \subseteq T_{\Sigma,s}$), and
- $\sigma(t_1 \ldots t_n) \in T_{\Sigma,s}$ whenever $\sigma \in \Sigma_{s_1 \ldots s_n,s}$ and $t_i \in T_{\Sigma,s_i}$ for $i \in \{1,\ldots,n\}$.

The operations in Σ can be interpreted on T_Σ in the obvious manner, thus making it into a Σ-algebra 0_Σ. If T is equational, then its ground part (i.e., the set of pairs of terms without variables representing the set of the ground equations of T) is a congruence $\equiv_T$ on 0_Σ. Then 0_T is the quotient $0_\Sigma/_{\equiv_T}$, whose carriers are equivalence classes of Σ-terms under $\equiv_T$. If T is a pure rewriting theory then 0_T is a preorder model[7] whose carriers $(0_T)_s$ are preorders of Σ-terms with the preorder relation given by the existence of a rewrite sequence (using the rules of T). Finally, rewrite theories including equations require the combination between the above two constructions.

Example 2 Consider the following CafeOBJ specification of non-deterministic natural numbers:

```
mod! NNAT {
  protecting(NAT)
  [ Nat < NNat ]
  op _|_ : NNat NNat -> NNat {assoc}
  trans M:Nat | N:Nat => M .
  trans M:Nat | N:Nat => N .
}
```

The denotation of NNAT is initial and consists [of the isomorphism class] of one model, $0_{\mathtt{NNAT}}$, the initial model. The main carrier of $0_{\mathtt{NNAT}}$ is a preorder of non-empty lists of natural numbers with the deletion sequences as the preorder relation. _|_ gets interpreted as a preorder functor which concatenates lists of numbers, and composes in parallel ("horizontally") deletion sequences.

[7] A restricted form of rewriting model; see the subsection "Models" of Section 2.2 for the definition.

We devote this section to the definition of the HOSRWL institution (defined for the first time in [9] in the many sorted version HRWL) which embeds all CafeOBJ cube institutions. We recall here that the behavioural specification part of HOSRWL is based on the 'coherent hidden algebra' of [14]. The deep understanding of HOSRWL requires further reading on its main components ([36] for RWL and [14] for CHA) as well as their integration [9].

Signatures

Definition 3 *A HOSRWL **signature** is a tuple* $(H,V,\leq,\Sigma,\Sigma^{\mathrm{b}})$*, where*

- $(H,\leq)$ *and* $(V,\leq)$ *are disjoint partially ordered sets of **hidden** sorts and **visible** sorts, respectively,*
- Σ *is a* $(H\cup V,\leq)$*-order-sorted signature,*
- $\Sigma^{\mathrm{b}}\subseteq\Sigma$ *is a subset of **behavioural operations** such that* $\sigma\in\Sigma^{b}_{w,s}$ *has* exactly *one hidden sort in w.*

Notice that we may simplify the notation $(H,V,\leq,\Sigma,\Sigma^{\mathrm{b}})$ to just $(H,V,\leq,\Sigma)$, or just Σ, when no confusion is possible.

Also notice that the CafeOBJ RWL signatures are just ordinary algebraic (MSA) signatures; our approach is thus rather different from the original definition of RWL signatures [36] adding structural equations in the definition of the signature.

From a methodological perspective, the operations in Σ^{b} have object-oriented meaning, $\sigma\in\Sigma^{\mathrm{b}}_{w,s}$ is thought of as an **action** (or "method" in a more classical jargon) on the space (type) of states if s is hidden, and thought of as and **observation** (or "attribute" in a more classical jargon) if s is visible. The last condition says that the actions and observations act on (states of) single objects.

Definition 4 *A HOSRWL **signature morphism*** $\Phi\colon (H,V,\leq,\Sigma,\Sigma^{\mathrm{b}})\to(H',V',\leq',\Sigma',\Sigma'^{\mathrm{b}})$ *is an order-sorted signature morphism* $(H\cup V,\leq,\Sigma)\to(H'\cup V',\leq',\Sigma')$ *such that*

(M1) $\Phi(V)\subseteq V'$ *and* $\Phi(H)\subseteq H'$*,*
(M2) $\Phi(\Sigma^{\mathrm{b}})=\Sigma'^{\mathrm{b}}$ *and* $\Phi^{-1}(\Sigma'^{\mathrm{b}})\subseteq\Sigma^{\mathrm{b}}$*,*
(M3) if $\Phi(h)<\Phi(h')$ *for any hidden sorts* $h,h'\in H$*, then* $h<h'$*.*

These conditions say that hidden sorted signature morphisms preserve visibility and invisibility for both sorts and operations, and the $\Sigma'^{\mathrm{b}}\subseteq\Phi(\Sigma^{\mathrm{b}})$ inclusion together with (M3) expresses the encapsulation of classes (in the sense that no new actions

(methods) or observations (attributes) can be defined on an imported class) [8] . However, these conditions apply only to the case when signature morphisms are used as module imports (the so-called *horizontal* signature morphisms); when they model specification refinement this condition might be dropped (this case is called *vertical* signature morphism).

Proposition 5 *HOSRWL signatures and signature morphisms (with the obvious composition) form a category denoted as* $\mathbb{S}ign^{\mathrm{HOSRWL}}$.

Sentences

In HOSRWL there are several kinds of sentences inherited from the various CafeOBJ cube institutions.

Definition 6 *Consider a HOSRWL signature* $(H, V, \leq, \Sigma, \Sigma^{\mathrm{b}})$. *Then a* ***(strict) equation*** *is a sentence of the form*

$$(\forall X)\ t\ =\ t'\ \textbf{if}\ C$$

where X *is a* $(H \cup V)$*-sorted set of variables,* t, t' *are* Σ*-terms with variables* X*, and* C *is a Boolean* [9] *(-sorted)* Σ*-term,*
a ***behavioural equation*** *is a sentence of the form*

$$(\forall X)\ t\ \sim\ t'\ \textbf{if}\ C,$$

a ***(strict) transition*** *is a sentence of the form*

$$(\forall X)\ t\ =>\ t'\ \textbf{if}\ C,$$

and a ***behavioural transition*** *is a sentence of the form*

$$(\forall X)\ t\ \sim>\ t'\ \textbf{if}\ C$$

where X, t, t', C *have the same meaning as for strict equations.*

All these sentences are here defined in the conditional form. If the condition is missing (which is equivalent to saying that it is always true), then we get the unconditional versions of sentences. Notice also that our approach to conditional sentences is slightly different from other approaches in the literature in the sense that the condition is a Boolean term rather than a finite conjunction of formulæ. Our

[8] Without it the Satisfaction Condition fails; for more details on the logical and computational relevance of this condition see [23].

[9] We implicitly assume the existence of a Boolean sort `Bool` together with the ordinary Boolean operations and equations specifying a Boolean data type `BOOL`.

approach is more faithful to the concrete level of CafeOBJ and is also more general. This means that a finite conjunction of formulæ can be translated to a Boolean term by using some special semantic predicates (such as == for semantic equality and ==> for the semantic transition relation, in CafeOBJ). We do not discuss here the full details of this approach, we only mention that the full rigorous treatment of such conditions can be achieved within the so-called *constraint logic* [11], which can however be regarded as a special case of an abstract categorical form of plain equational logic [6, 5, 11].

Equational attributes such as associativity (A), commutativity (C), identity (I), or idempotence (Z) are just special cases of strict equations. However, the behavioural part of HOSRWL has another special attribute called *behavioural coherence* [13, 14] which is regarded as a sentence:

Definition 7 *Let* $(H, V, \leq, \Sigma, \Sigma^{\mathrm{b}})$ *be a signature. Then*

σ **coherent**

is a ***behavioural coherence*** *declaration for* σ*, where* σ *is any operation* Σ*.*

Definition 8 *Given a signature morphism* $\Phi\colon (H, V, \leq, \Sigma, \Sigma^{\mathrm{b}}) \to (H', V', \leq', \Sigma', \Sigma'^{\mathrm{b}})$ *the translation of sentences is defined by replacing all operation symbols from* Σ *with the corresponding symbols (via* Φ*) from* Σ' *and by re-arranging the sort of the variables involved accordingly to the sort mapping given by* Φ*.*

Fact 9 *If we denote the set of sentences of a signature* $(H, V, \leq, \Sigma, \Sigma^{\mathrm{b}})$ *by* $Sen^{\mathrm{HOSRWL}}(H, V, \leq, \Sigma, \Sigma^{\mathrm{b}})$ *and the sentence translation corresponding to a signature morphism* Φ *by* $Sen^{\mathrm{HOSRWL}}(\Phi)$*, then we get a sentence functor* $Sen^{\mathrm{HOSRWL}}\colon \mathbb{S}ign^{\mathrm{HOSRWL}} \to \mathbb{S}et$*.*

Models

Models of HOSRWL are *preorder models* which are (algebraic) interpretations of the signatures into $\mathbb{P}re$ (the category of preorders) rather than in $\mathbb{S}et$ (the category of sets) as in the case of ordinary algebras. Thus, ordinary algebras can be regarded as a special case of preorder models with discrete carriers. On the other hand, if we ignore the order sorted aspect, the HOSRWL preorder models are a special case of Meseguer RWL models [36] which have at most one arrow between elements. In the case of preorder models, the arrows between elements are called *transitions*.

Definition 10 *Given a HOSRWL signature* $(H, V, \leq, \Sigma, \Sigma^{\mathrm{b}})$*, a HOSRWL* ***model*** *M interprets:*

- *each sort s as a preorder* M_s *and each subsort relation* $s < s'$ *as a sub-category relation* $M_s \subseteq M_{s'}$*, and*

- *each operation $\sigma \in \Sigma_{w,s}$ as a preorder functor $M_\sigma : M_w \to M_s$, where M_w stands for $M_{s_1} \times \ldots \times M_{s_n}$ for $w = s_1 \ldots s_n$.*

Notice that each Σ-term $t : w \to s$ gets an associated preorder functor $M_t : M_w \to M_s$ by evaluating it for each assignment of the variables occurring in t with elements from the corresponding carriers of M.

Model homomorphisms in HOSRWL follow an idea of [31] by refining the ordinary concept of model morphism and reforming the hidden algebra [23, 24] homomorphisms by taking adequate care of the behavioural structure of models. We need first to define the concept of *behavioural equivalence*.

Definition 11 *Recall that a Σ-**context** $c[z]$ is any Σ-term c with a marked variable z occurring only once in c. A context $c[z]$ is **behavioural** iff all operations above* [10] *z are behavioural.*

*Given a model M, two elements (of the same sort s) a and a' are called **behaviourally equivalent**, denoted $a \sim_s a'$ (or just $a \sim a'$) iff* [11]

$$M_c(a) = M_c(a')$$

for all visible *behavioural contexts c.*

Remark that the behavioural equivalence is a $(H \cup V)$-sorted equivalence relation, and on the visible sorts the behavioural equivalence coincides with the (strict) equality relation.

Now we are ready to give the definition of model homomorphism in HOSRWL.

Definition 12 *A homomorphism $h : M \to M'$ between models of a signature $(H, V, \leq, \Sigma, \Sigma^{\mathrm{b}})$ is a $(H \cup V)$-sorted categorical relation* [12] *between the preorder carriers such that (for each sort s):*

- *for all $a \in M_s$ there exists $a' \in M'_s$ (where a and a' can be either both transitions or both elements) such that $a \; h_s \; a'$,*
- *for all $a \in M_s$, if $a \; h_s \; a'$ then ($a \; h_s \; b'$ if and only if $a' \sim_s b'$),*
- *for all $a, b \in M_s$ and $a' \in M'_s$, if $a \; h_s \; a'$ and $a \sim_s b$ then $b \; h_s \; a'$, and*
- *for each operation $\sigma \in \Sigma_{w,s}$, for all $a \in M_w$ and $a' \in M'_w$, $a \; h_w \; a'$ (componentwise) implies $M_\sigma(a) \; h_s \; M'_\sigma(a')$.*

[10] Meaning that z is in the subterm determined by the operation.

[11] Notice that this equality means an equality between functors $M_{w_1 w_2} \to M_{s'}$, where $c : w_1 s w_2 \to s'$ with $w_1, w_2 \in (H \cup V)^*$ and $s' \in V$.

[12] This is a relation between the sets of elements together with a relation between the sets of transitions, such that this couple of relations commute with the domain functions, codomain functions, and transition composition functions.

Notice that when there are no hidden sorts (i.e., we are in some non-behavioural part of HOSRWL), this concept of model homomorphism coincides with the rewriting model homomorphism.

For a given signature $(H,V,\leq,\Sigma)$, we denote its category of models by $\mathrm{MOD}^{\mathrm{HOSRWL}}$. Notice that any signature morphism $\Phi\colon (H,V,\leq,\Sigma,\Sigma^{\mathrm{b}}) \to (H',V',\leq',\Sigma',\Sigma'^{\mathrm{b}})$ determines a *model reduct functor* $\mathrm{MOD}(\Phi)\colon \mathrm{MOD}(H',V',\leq',\Sigma',\Sigma'^{\mathrm{b}}) \to \mathrm{MOD}(H,V,\leq,\Sigma,\Sigma^{\mathrm{b}})$ in the usual way (by renaming the sorts of the carriers and the interpretations of the operations accordingly to the mapping of sorts and operations given by Φ). Therefore we have a contravariant *model functor*
$\mathrm{MOD}^{\mathrm{HOSRWL}}\colon \mathbb{S}ign^{\mathrm{HOSRWL}} \to \mathbb{C}at^{\mathrm{op}}$.

Satisfaction

The satisfaction relation between sentences and models is the crucial concept of an institution (see Definition 20).

Definition 13 *Consider a model M of a signature $(H,V,\leq,\Sigma,\Sigma^{\mathrm{b}})$. Then M* ***satisfies an equation****, i.e., $M \models (\forall X)\ t = t'$* **if** *$C$, if and only if*

$$M_t(\theta) = M_{t'}(\theta) \ \textit{ whenever } \ M_C(\theta) \ \textit{ is true}$$

for all valuations $\theta\colon X \to M$. (Notice that we implicitly assume the standard (initial) interpretation by M of the built-in data type BOOL.)

M **satisfies a behavioural equation**, i.e., *$M \models (\forall X)\ t \sim t'$* **if** *$C$*, if and only if

$$M_t(\theta) \sim M_{t'}(\theta) \ \text{ whenever } \ M_C(\theta) \ \text{ is true}$$

for all valuations $\theta\colon X \to M$.

M **satisfies a transition**, i.e., *$M \models (\forall X)\ t => t'$* **if** *$C$*, if and only if for each valuation $\theta\colon X \to M$ there exists the transition $M_t(\theta) \to M_{t'}(\theta)$ whenever $M_C(\theta)$ is true.

M **satisfies a behavioural transition**, i.e., *$M \models (\forall X)\ t \sim> t'$* **if** *$C$*, if and only if for each appropriate visible behavioural context c and for each valuation $\theta\colon X \to M$ there exists the transition $M_c(M_t(\theta)) \to M_c(M_{t'}(\theta))$ whenever $M_C(\theta)$ is true.

Finally, *M* **satisfies a coherence declaration**, i.e., *$M \models (\sigma$* **coherent** *)*, if and only if σ preserves the behavioural equivalence on M, i.e.,

$$M_\sigma(a) \sim M_\sigma(a') \ \text{ if } \ a \sim a' \ \text{ (component-wise)}$$

for all $a,a' \in M_w$.

Notice that the behavioural coherence of both the behavioural operations and of operations of a visible rank is trivially satisfied.

Example 14 Consider the following CafeOBJ behavioural specification of non-deterministic natural numbers:

```
mod* NNAT-HSA {
  protecting(NAT)
  *[ NNat ]*
  op [_] : Nat -> NNat
  op _|_ : NNat NNat -> NNat
  bop _->_ : NNat Nat -> Bool
  vars S1 S2 : NNat
  vars M N : Nat
  eq [M] -> N = M == N .
  eq S1 | S2 -> N = S1 -> N or S2 -> N .
}
```

The non-deterministic natural numbers s_1 and s_2 are behaviourally equivalent if and only if

$$s_1 \to n \text{ is true if and only if } s_2 \to n \text{ is true}$$

for all natural numbers n.

Notice that for all models M of NNAT-HSA,

$$M \models (\_|\_ \textbf{ coherent })$$

This situation where the operations which are neither behavioural nor data type operations (i.e. with visible rank) are automatically coherent is rather natural and occurs very often in practice, and this corresponds to the so-called *coherence conservative methodology* of [14].

The definition of the satisfaction relation between sentences and models completes the construction of the HOSRWL institution:

Theorem 15 $(\mathbb{S}ign^{\text{HOSRWL}}, Sen^{\text{HOSRWL}}, \text{MOD}^{\text{HOSRWL}}, \models)$ *is an institution.*

For the definition of institution see Definition 20 given below. We omit here the proof of this result which is rather long and tedious and follows the same pattern as proofs of similar results, also reusing some of them.

At the end of the presentation of the HOSRWL institution we give a brief example of a CafeOBJ specification in HOSRWL:

Example 16 Consider a behavioural specification of sets of non-deterministic natural numbers:

```
mod* SETS {
  protecting(NNAT)
  *[ Set ]*
  op empty : -> Set
  op add   : NNat Set -> Set   {coherent}
  op _U_ : Set Set -> Set      {coherent}
  op _&_ : Set Set -> Set      {coherent}
  op not : Set -> Set          {coherent}
  bop _in_ : NNat Set -> Bool
  vars E E' : NNat
  vars S S1 S2 : Set
  eq E in empty = false .
  eq E in add(E', S) = (E == E') or (E in S) .
  eq E in S1 U S2  = (E in S1)  or (E in S2) .
  eq E in S1 & S2  = (E in S1) and (E in S2) .
  eq E in not(S1)  = not (E in S1) .
}
```

where NNAT is the RWL specification of non-deterministic natural numbers of Example 2. Models of SETS interpret the non-deterministic naturals NNAT by the initial model $0_{\texttt{NNAT}}$ and the hidden sort Set and its related operations in various ways. One possible way is to interpret Set as sets of non-deterministic naturals. Another way would be to interpret Set as a pair between a Boolean element and a list of non-deterministic naturals, with the Boolean false playing the role of negation and with a corresponding interpretation of the operations. There are also many other ways to interpret the loose part of SETS. Notice that each model of SETS satisfies the usual set theory rules (such as commutativity and associativity of union and intersection, De Morgan laws, etc.) only *behaviourally*, not necessarily in the strict sense. For example, the behavioural commutativity of the union

```
  beq S1 U S2 = S2 U S1 .
```

is a consequence of the specification SETS. While the former model satisfies it strictly, the latter does not when interpreting the union as list concatenation.

Specifications in full HOSRWL naturally occur in the case of a behavioural specification using concurrent (RWL) data types. However the practical significance of full HOSRWL is still little understood. The real importance of the HOSRWL institution is its initiality in the CafeOBJ cube. We will see below that the existence of all possible combinations between the main logics/institutions of CafeOBJ is crucial for the good properties of the CafeOBJ institution.

The operational semantics underlies the execution of specifications or programs. As with OBJ, the CafeOBJ operational semantics is based on rewriting, which in the case of proofs is used without directly involving the user defined transitions (rules) as rewrite rules but rather involving them via the built-in semantic transition predicate ==>. [13] For executions of concurrent systems specified in rewriting logic, CafeOBJ uses both the user-defined transitions and equations.

Since rewriting is a very well known topic in algebraic specification, we do not insist here on the standard aspects of rewriting. However, the operational semantics of behavioural specification requires a more sophisticated notion of rewriting which takes special care of the use of behavioural sentences during the rewriting process, which we call *behavioural rewriting* [13, 14]:

Definition 17 *Given a HOSRWL signature Σ and a Σ-algebra A, a **behaviourally coherent context for** A is any Σ-context $c[z]$ such that all operations above the marked variable z are either behavioural or behaviourally coherent for A.*

Notice that any behavioural context is also behaviourally coherent.

Definition 18 *Consider a HOSRWL signature Σ, a set E of Σ-sentences regarded as a TRS (i.e. term rewriting system), and a Σ-algebra A satisfying the sentences in E. If t_0 is a ground term, then any rewrite step $t_0 \to t_1$ which uses a behavioural equation from E and for which the rewrite context has a behaviourally coherent sub-context for A is called a **behavioural rewriting step**.*

The following Proposition from [14] ensures the soundness of behavioural rewriting:

Proposition 19 *Under the hypotheses of Definition 18, if $t_0 \to t_1$ is a behavioural rewrite step, then $A \models (\forall \emptyset)\ t_0 \sim t_1$. Moreover, if the rewrite context is visible, then $A \models (\forall \emptyset)\ t_0 = t_1$.*

The completeness of the operational semantics with respect to the logical semantics is a two-layer completeness going via the important intermediate level of the proof calculi.

The completeness of the proof calculus is one of the most important class of results in algebraic specification, for equational logic we refer to [27], and for rewriting

[13] This means that the CafeOBJ proofs are equational and involve a built-in equation $(t ==> t') = \texttt{true}$ for each user defined transition $t => t'$. See [13] or [16] for more details.

logic to [36]. In the case of rewriting logic the relationship between the proof calculus and rewriting is very intimate, but for equational logic the completeness of rewriting can be found, among other many places, in [20, 8].

Notice that hidden logics of the CafeOBJ cube do not admit a complete (finitary) proof calculus [2]. However, advanced proof techniques support the verification process in the case of behavioural specifications, most notably the *hidden coinduction* method (see [24] for the original definition, [13, 14] for its realization in CafeOBJ, and [7] for the details for the case of proving behavioural transitions).

3 The CafeOBJ Institution

In this section we define the CafeOBJ institution, which is a Grothendieck construction on the CafeOBJ cube. The Grothendieck construction for institutions was introduced and developed by Diaconescu in [12] and generalises the famous Grothendieck construction for categories [30]. The essence of this Grothendieck construction is that it constructs a 'disjoint sum' of all institutions of the CafeOBJ cube, also introducing theory morphisms across the institution embeddings of the CafeOBJ cube. Such *extra theory morphisms* were first studied in [10]. However, one advantage of the Grothendieck institutions is that they treat the extra theory morphisms as ordinary theory morphisms, thus leading to a conceptual simplification with respect to [10].

The reader might wonder why one cannot live with HOSRWL only (which embeds all the CafeOBJ cube institutions) and we still need a Grothendieck construction on the CafeOBJ cube. The reason for this is that the combination of logics/institutions realized by HOSRWL collapses crucial semantic information, therefore a more refined construction which preserves the identity of each of the CafeOBJ cube institutions, but yet allowing a concept of theory morphism across the institution embeddings, is necessary. For example, in the case of specifications with loose semantics without a RWL component, the carriers of the models of these specifications should be sets rather than preorders, which is not possible in HOSRWL. Therefore, such specifications should be given semantics within the appropriate institution of the CafeOBJ cube rather than in HOSRWL. Example 39 illustrates this argument.

3.1 Institutions

We now recall from [21] the definitions of the main institution concepts:

Definition 20 *An **institution*** $\mathfrak{I} = (\mathbb{S}ign, Sen, \text{MOD}, \models)$ *consists of*

*(1) a category $\mathbb{S}ign$, whose objects are called **signatures**,*
*(2) a functor $Sen\colon \mathbb{S}ign \to \mathbb{S}et$, giving for each signature a set whose elements are called **sentences** over that signature,*
*(3) a functor $\mathrm{MOD}\colon \mathbb{S}ign^{\mathrm{op}} \to \mathbb{C}at$ giving for each signature Σ a category whose objects are called Σ-**models**, and whose arrows are called Σ-**(model) morphisms**, and*
*(4) a relation $\models_\Sigma \subseteq |\mathrm{MOD}(\Sigma)| \times Sen(\Sigma)$ for each $\Sigma \in |\mathbb{S}ign|$, called Σ-**satisfaction**,*

*such that for each morphism $\varphi\colon \Sigma \to \Sigma'$ in $\mathbb{S}ign$, the **satisfaction condition***

$$m' \models_{\Sigma'} Sen(\varphi)(e) \;\; \textit{iff} \;\; \mathrm{MOD}(\varphi)(m') \models_\Sigma e$$

holds for each $m' \in |\mathrm{MOD}(\Sigma')|$ and $e \in Sen(\Sigma)$. We may denote the reduct functor $\mathrm{MOD}(\varphi)$ by $\_\restriction_\varphi$ and the sentence translation $Sen(\varphi)$ by $\varphi(\_)$.

Definition 21 *Let $\mathfrak{I} = (\mathbb{S}ign, Sen, \mathrm{MOD}, \models)$ be an institution. For any signature Σ the closure of a set E of Σ-sentences is $E^\bullet = \{e \mid E \models_\Sigma e\}$* [14]*. (Σ, E) is a **theory** if and only if E is closed, i.e., $E = E^\bullet$.*

*A **theory morphism** $\varphi\colon (\Sigma, E) \to (\Sigma', E')$ is a signature morphism $\varphi\colon \Sigma \to \Sigma'$ such that $\varphi(E) \subseteq E'$. Let $\mathbb{T}h(\mathfrak{I})$ denote the category of all theories in $\mathfrak{I}$.*

For any institution $\mathfrak{I}$, the model functor MOD extends from the category of its signatures $\mathbb{S}ign$ to the category of its theories $\mathbb{T}h(\mathfrak{I})$, by mapping a theory (Σ, E) to the full subcategory $\mathrm{MOD}(\Sigma, E)$ of $\mathrm{MOD}(\Sigma)$ formed by the Σ-models which satisfy E.

Definition 22 *A theory morphism $\varphi\colon (\Sigma, E) \to (\Sigma', E')$ is **liberal** if and only if the reduct functor $\_\restriction_\varphi\colon \mathrm{MOD}(\Sigma', E') \to \mathrm{MOD}(\Sigma, E)$ has a left-adjoint $(\_)^\varphi$.*

*The institution $\mathfrak{I}$ is **liberal** if and only if each theory morphism is liberal.*

Definition 23 *An institution $\mathfrak{I} = (\mathbb{S}ign, Sen, \mathrm{MOD}, \models)$ is **exact** if and only if the model functor $\mathrm{MOD}\colon \mathbb{S}ign^{\mathrm{op}} \to \mathbb{C}at$ preserves finite limits. $\mathfrak{I}$ is **semi-exact** if and only if MOD preserves pullbacks.*

Definition 24 *Let $\mathfrak{I}$ and $\mathfrak{I}'$ be institutions. Then an **institution homomorphism** $\mathfrak{I}' \to \mathfrak{I}$ consists of*

(1) a functor $\Phi\colon \mathbb{S}ign' \to \mathbb{S}ign$,
(2) a natural transformation $\alpha\colon \Phi; Sen \Rightarrow Sen'$, and
(3) a natural transformation $\beta\colon \mathrm{MOD}' \Rightarrow \Phi^{\mathrm{op}}; \mathrm{MOD}$

[14] $E \models_\Sigma e$ means that $M \models_\Sigma e$ for any Σ-model M that satisfies all sentences in E.

such that the following ***satisfaction condition*** *holds*

$$m' \models_{\Sigma'} \alpha_{\Sigma'}(e) \;\; \textit{iff} \;\; \beta_{\Sigma'}(m') \models'_{\Sigma'\Phi} e$$

for any Σ'-model m' from $\mathfrak{I}'$ and any $\Sigma'\Phi$-sentence e from $\mathfrak{I}$.

Fact 25 *Institutions and institution homomorphisms form a category denoted as $\mathbb{I}ns$.*

The following properties of institution homomorphisms were defined in [12] and play an important rôle for Grothendieck institutions:

Definition 26 *An institution homomorphism $(\Phi, \alpha, \beta) : \mathfrak{I}' \to \mathfrak{I}$ is*

- *an* ***embedding*** *iff Φ admits a left-adjoint $\overline{\Phi}$ (with unit ζ); an institution embedding is denoted as $(\Phi, \overline{\Phi}, \zeta, \alpha, \beta) : \mathfrak{I}' \to \mathfrak{I}$, and is*
- ***liberal*** *iff $\beta_{\Sigma'}$ has a left-adjoint $\overline{\beta}_{\Sigma'}$ for each $\Sigma' \in |\mathbb{S}ign'|$.*

An institution embedding $(\Phi, \overline{\Phi}, \zeta, \alpha, \beta) : \mathfrak{I}' \to \mathfrak{I}$ is ***exact*** *if and only if the square below is a pullback*

$$\begin{array}{ccc}
\textsc{Mod}(\Sigma) & \xleftarrow{\textsc{Mod}(\varphi)} & \textsc{Mod}(\Sigma_1) \\
{\scriptstyle \textsc{Mod}(\Sigma\zeta)}\uparrow & & \uparrow{\scriptstyle \textsc{Mod}(\Sigma_1\zeta)} \\
\textsc{Mod}(\Sigma\overline{\Phi}\Phi) & & \textsc{Mod}(\Sigma_1\overline{\Phi}\Phi) \\
{\scriptstyle \beta_{\Sigma\overline{\Phi}}}\uparrow & & \uparrow{\scriptstyle \beta_{\Sigma_1\overline{\Phi}}} \\
\textsc{Mod}'(\Sigma\overline{\Phi}) & \xleftarrow[\textsc{Mod}'(\varphi\overline{\Phi})]{} & \textsc{Mod}'(\Sigma_1\overline{\Phi})
\end{array}$$

where $\varphi : \Sigma \to \Sigma_1$ is any signature morphism in $\mathfrak{I}$.

3.2 Indexed and Grothendieck Institutions

The following definition from [12] generalises the concept of indexed category [40] to institutions.

Definition 27 *An* ***indexed institution*** *$\mathfrak{I}$ is a functor $\mathfrak{I} : I^{\mathrm{op}} \to \mathbb{I}ns$.*

The CafeOBJ cube is an indexed institution where the index category I is the 8-element lattice corresponding to the cube (i.e., the elements of the lattice correspond to the nodes of the cube and the partial order is given by the arrows of the cube).

Definition 28 *The* ***Grothendieck institution*** *$\mathfrak{I}^\sharp$ of an indexed institution $\mathfrak{I} : I^{\mathrm{op}} \to$*

$\mathbb{I}ns$ has

(1) the Grothendieck category $\mathbb{S}ign^\sharp$ as its category of signatures, where $\mathbb{S}ign\colon I^{\mathrm{op}} \to \mathbb{C}at$ is the indexed *category of signatures of the indexed institution $\mathfrak{I}$,*

(2) $\mathrm{MOD}^\sharp\colon (\mathbb{S}ign^\sharp)^{\mathrm{op}} \to \mathbb{C}at$ as its model functor, where

- *$\mathrm{MOD}^\sharp(\langle i, \Sigma\rangle) = \mathrm{MOD}^i(\Sigma)$ for each index $i \in |I|$ and signature $\Sigma \in |\mathbb{S}ign^i|$, and*
- *$\mathrm{MOD}^\sharp(\langle u, \varphi\rangle) = \beta^u_\Sigma; \mathrm{MOD}^i(\varphi)$ for each $\langle u, \varphi\rangle\colon \langle i, \Sigma\rangle \to \langle i', \Sigma'\rangle$,*

(3) $Sen^\sharp\colon \mathbb{S}ign^\sharp \to \mathbb{S}et$ as its sentence functor, where

- *$Sen^\sharp(\langle i, \Sigma\rangle) = Sen^i(\Sigma)$ for each index $i \in |I|$ and signature $\Sigma \in |\mathbb{S}ign^i|$, and*
- *$Sen^\sharp(\langle u, \varphi\rangle) = Sen^i(\varphi); \alpha^u_{\Sigma'}$ for each $\langle u, \varphi\rangle\colon \langle i, \Sigma\rangle \to \langle i', \Sigma'\rangle$,*

(4) $m \models^\sharp_{\langle i,\Sigma\rangle} e$ iff $m \models^i_\Sigma e$ for each index $i \in |I|$, signature $\Sigma \in |\mathbb{S}ign^i|$, model $m \in |\mathrm{MOD}^\sharp(\langle i, \Sigma\rangle)|$, and sentence $e \in Sen^\sharp(\langle i, \Sigma\rangle)$.

where $\mathfrak{I}^i = (\mathbb{S}ign^i, \mathrm{MOD}^i, Sen^i, \models^i)$ for each index $i \in |I|$ and $\mathfrak{I}^u = (\Phi^u, \alpha^u, \beta^u)$ for $u \in I$ index morphism.

For the category minded readers we mention that [12] gives a higher level characterisation of the Grothendieck institution as a lax colimit in the 2-category $\mathbb{I}ns$ (with institutions as objects, institution homomorphisms as 1-cells, and institution *modifications* as 2-cells; see [12] for details) of the corresponding indexed institution. This means that Grothendieck institutions are internal *Grothendieck objects* [15] in $\mathbb{I}ns$ in the same way as Grothendieck categories are Grothendieck objects in $\mathbb{C}at$. For the fibred category minded readers, [12] also introduces the alternative formulation of *fibred institution* and shows that there is a natural equivalence between split fibred institutions and Grothendieck institutions.

We would also like to mention that the concept of extra theory morphism [10] across an institution homomorphism $\mathfrak{I}' \to \mathfrak{I}$ (with all its subsequent concepts) is recovered as an ordinary theory morphism in the Grothendieck institution of the indexed institution given by the homomorphism $\mathfrak{I}' \to \mathfrak{I}$ (i.e., which has $\bullet \to \bullet$ as its index category).

Now we are ready to define the institution of CafeOBJ:

Definition 29 *The CafeOBJ **institution** is the Grothendieck institution of the CafeOBJ cube.*

[15] From [12], a Grothendieck object in a 2-category is a lax colimit of a 1-functor to that 2-category.

3.3 Properties of the CafeOBJ Institution

In this section, we briefly study the most important institutional properties of the CafeOBJ institution: existence of theory colimits, liberality (i.e. free constructions), and exactness (i.e. model amalgamation).

Proposition 30 *The institution homomorphisms of the CafeOBJ cube are all embeddings.*

Sketch of Proof: The forgeful functors between the categories of the signatures of the CafeOBJ institutions are as follows:

- The forgetful functors along the order sorted dimension forget the ordinary sorts, i.e., a signature $(S, \leq, \Sigma)$ gets mapped to (S, Σ). The left-adjoints to these functors map a signature (S, Σ) to the discrete order sorted signature $(S, =, \Sigma)$.
- The forgetful functors along the RWL dimension are all identities, so they trivially admit left-adjoints.
- The forgetful functors along the behavioural dimension forget the hidden sorts and the operations involving the hidden sorts. Thus, a signature $(H, V, \Sigma, \Sigma^{\mathrm{b}})$ gets mapped to (V, Σ^V) where Σ^V is the set of operations in Σ having only visible sorts in the rank (that is, involving only visible sorts). The left-adjoints to these functors map a signature (S, Σ) to the behavioural signature $(\emptyset, S, \Sigma, \emptyset)$.

This makes the CafeOBJ cube an *embedding-indexed institution* (cf. [12]). As we will see below, this property of the CafeOBJ cube plays an important rôle for the properties of the CafeOBJ institution.

Theory Colimits.

The existence of theory colimits is crucial for any module system in the Clear-OBJ tradition. Let us recall the following result from [12]:

Theorem 31 *Let $\mathfrak{I} : I^{\mathrm{op}} \to \mathbb{I}ns$ be an embedding-indexed institution such that I is J-cocomplete for a small category J. Then the category of theories $\mathbb{T}h(\mathfrak{I}^\sharp)$ of the Grothendieck institution $\mathfrak{I}^\sharp$ has J-colimits if and only if the category of signatures $\mathbb{S}ign^i$ is J-cocomplete for each index $i \in |I|$.*

Corollary 32 *The category of theories of the CafeOBJ institution is small cocomplete.*

Notice that the fact that the lattice of institutions of the CafeOBJ cube is complete (as a lattice) means exactly that the index category of the CafeOBJ cube is (small) cocomplete, which is a precondition for the existence of theory colimits in the CafeOBJ institution. In the absence of the combinations of logics/institutions

of the CafeOBJ cube (such as HOSRWL), the possibility of theory colimits in the CafeOBJ institution would have been lost.

Liberality.

Liberality is a desirable property in relation to initial denotations for structured specifications. In the case of loose denotations liberality is not necessary. Since the behavioural specification paradigm involves only loose denotations, in the case of the CafeOBJ institution, we are therefore interested in liberality only for the non-behavioural theories. Recall the following result from [12]:

Theorem 33 *The Grothendieck institution $\mathfrak{I}^\sharp$ of an indexed institution $\mathfrak{I} : I^{\mathrm{op}} \to \mathbb{I}ns$ is liberal if and only if $\mathfrak{I}^i$ is liberal for each index $i \in |I|$ and each institution homomorphism $\mathfrak{I}^u$ is liberal for each index morphism $u \in I$.*

Corollary 34 *In the CafeOBJ institution, each theory morphism between non-behavioural theories is liberal.*

This corollary is obtained from the theorem above by restricting the index category to the non-behavioural square of the CafeOBJ cube, and from the corresponding liberality results for equational and rewriting logics (see [32] for a general liberality result which instantiate to the CafeOBJ equational and rewriting logics [16]).

Exactness.

Firstly, let us extend the well known exactness results for equational logic [17] to the CafeOBJ cube: [17]

Proposition 35 *All institutions of the CafeOBJ cube are semi-exact.*

Notice that the exactness of the hidden (behavioural) logics can be deduced directly from the exactness of the equational and rewriting logics because the models of the hidden logics are essentially just ordinary algebras or preorder models.

As shown in [10] and [12], in practice exactness is a property hardly achieved at the global level by the Grothendieck institutions. In [12] we give a necessary and sufficient set of conditions for (semi-)exactness of Grothendieck institutions. One of them is the exactness of the institution embeddings, which fails for the embeddings from the non-RWL institutions into the RWL institutions of the CafeOBJ cube as shown by the following:

[16] For the liberality of Meseguer RWL the reader might look into [35].

[17] For the exactness of rewriting logics we may refer to Hendrik Hilberdink coming Oxford DPhil thesis; [32] is a condensed version of some parts of it.

Fact 36 *The embedding of MSA into RWL is* not *exact.*

Proof: We do a proof by contradiction. Let us assume that the embedding of MSA into RWL is exact. For the signature homomorphism φ of Definition 26 we choose the unique signature homomorphism $\varphi\colon \emptyset \to \Sigma_1$ from the empty signature $\emptyset$ to an arbitrary but fixed signature Σ_1.

The pullback square of Definition 26 gets simplified to the fact that the category $\text{MOD}^{\text{RWL}}(\Sigma_1)$ is the product of the categories $\text{MOD}^{\text{RWL}}(\emptyset)$ and $\text{MOD}^{\text{MSA}}(\Sigma_1)$. But $\text{MOD}^{\text{RWL}}(\emptyset)$ is the terminal category, thus we deduce that $\text{MOD}^{\text{RWL}}(\Sigma_1)$ and $\text{MOD}^{\text{MSA}}(\Sigma_1)$ are isomorphic, which is obviously wrong.

In the absence of a desired global exactness property for the CafeOBJ institution, we need a set of sufficient conditions for exactness for practically significant particular cases. In [10] we formulate a set of such sufficient conditions, but this problem is still open.

4 Foundations of Structured Specifications

In this section we survey the mathematical foundations of the CafeOBJ module composition system, which follows the principles of the OBJ module system which are inherited from earlier work on Clear [1]. Consequently, the CafeOBJ module system is institution-independent (i.e., can be developed at the abstract level of institutions) in the style of [17]. In the actual case of CafeOBJ, the institution-independent semantics is instantiated to the CafeOBJ institution. The following principle governs the semantics of programming in-the-large in CafeOBJ:

> **(L)** For each structured specification we consider the theory corresponding to its flattening to a basic specification. The structuring constructs are modelled as theory morphisms between appropriate theories. The denotation $[\![SP]\!]$ of a structured specification is determined from the denotations of the components recursively via the structuring constructs involved.

The general structuring mechanism is constituted by *module expressions*, which are iterations of several basic structuring operations, such as (multiple) imports, parameters, instantiation of parameters by views, translations, etc.

Module imports constitute the most primitive structuring construct in any module composition system. The concept of module import in the institution-independent semantics of CafeOBJ is based on the mathematical notion of *inclusion system*.

> Module imports are modelled as inclusion theory morphisms between the theories corresponding to flattening the imported and the importing modules.

Inclusion systems were first defined in [17] for the institution-independent study of structuring specifications. *Weak inclusion systems* were introduced in [4], and they constitute a simplification of the original definition of inclusion systems of [17]. We recall the definition of inclusion systems:

Definition 37 *$\langle I, \mathcal{E} \rangle$ is a **weak inclusion system** for a category $\mathbb{C}$ if I and $\mathcal{E}$ are two sub-categories with $|I| = |\mathcal{E}| = |\mathbb{C}|$ such that*

(1) I is a partial order, and
(2) every arrow f in $\mathbb{C}$ can be factored uniquely as $f = e; i$ with $e \in \mathcal{E}$ and $i \in I$.

*The arrows of I are called **inclusions**, and the arrows of $\mathcal{E}$ are called **surjections**.* [18] *The domain (source) of the inclusion i in the factorisation of f is called the **image of** f and denoted as* $\mathrm{Im}(f)$. *An **injection** is a composition between an inclusion and an isomorphism.*

*A weak inclusion system $\langle I, \mathcal{E} \rangle$ is an **inclusion system** iff I has finite least upper bounds (denoted $+$) and all surjections are epics (see [17]).*

The inclusion system for the category of theories of the CafeOBJ institution is obtained by lifting the inclusion system for its category of signatures (see [17, 4]). The weak inclusion system for the category of signatures is obtained from the canonical inclusion systems of the categories of signatures [19] of the CafeOBJ cube institutions by using the following result from [12] (which appeared previously in a slightly different form in [10]):

Theorem 38 *Let $B\colon I^{\mathrm{op}} \to \mathbb{C}at$ be an indexed category such that*

[18] Surjections of some weak inclusion systems need not necessarily be surjective in the ordinary sense.

[19] For example, in the simplest case of the MSA signatures, an *inclusion* $(S,\Sigma) \hookrightarrow (S',\Sigma')$ is given by $S \hookrightarrow S'$ and $\Sigma_{w,s} \hookrightarrow \Sigma'_{w,s}$ (as ordinary set-theoretic inclusions) for each $w \in S^*$ and $s \in S$. $(f,g)\colon (S,\Sigma) \to (S',\Sigma')$ is *surjection* iff $S' = f(S)$ and $\Sigma'_{w',s'} = \bigcup\{g(\Sigma_{w,s}) \mid f(w) = w' \text{ and } f(s) = s'\}$, for each w', s'. This example is originally developed in [17] and can be easily extended to the other more complex CafeOBJ cube institutions.

- *I has a weak inclusion system $\langle I^I, \mathcal{E}^I \rangle$,*
- *B^i has a weak inclusion system $\langle I^i, \mathcal{E}^i \rangle$ for each index $i \in |I|$,*
- *B^u preserves inclusions for each inclusion index morphism $u \in I^I$, and*
- *B^u preserves inclusions and surjections and lifts inclusions uniquely for each surjection index morphism $u \in \mathcal{E}^I$.*

Then, the Grothendieck category $B^\sharp$ has an inclusion system $\langle I^{B^\sharp}, \mathcal{E}^{B^\sharp} \rangle$ where $\langle u, \varphi \rangle$ is

- inclusion *iff both u and φ are inclusions, and*
- surjection *iff both u and φ are surjections.*

In the case of the CafeOBJ institution, this result is applied for the indexed category of signatures of the CafeOBJ cube (see Proposition 30 for details on the structure of the indexed category of signatures of the CafeOBJ cube).

Example 39 Consider the following module import:

```
mod* TRIV { [ Elt ] }

mod* NTRIV {
  protecting(TRIV)
  op _|_ : Elt Elt -> Elt {assoc}
  trans M:Elt | N:Elt => M .
  trans M:Elt | N:Elt => N .
}
```

Module `TRIV` gets a MSA loose theory, which has all sets as its denotation. Module `NTRIV` gets a RWL loose theory, which has as denotations preorders with an interpretation of _|_ as an associative binary preorder functor, and which satisfy the couple of choice transitions of `NTRIV`. The module import `TRIV` $\rightarrow$ `NTRIV` corresponds to an injective extra theory morphism $T^{\texttt{TRIV}} \rightarrow T^{\texttt{NTRIV}}$ across the forgetful institution morphism RWL $\rightarrow$ MSA.

More formally, the inclusion signature morphism underlying $T^{\texttt{TRIV}} \rightarrow T^{\texttt{NTRIV}}$ can be represented as $\langle u, \varphi \rangle$ where u is the institution morphism RWL $\rightarrow$ MSA and φ is the signature inclusion $\Sigma^{\texttt{TRIV}} \rightarrow u(\Sigma^{\texttt{NTRIV}})$ (where $\Sigma^{\texttt{TRIV}}$ is the MSA signature of `TRIV`, $\Sigma^{\texttt{NTRIV}}$ is the RWL signature of `NTRIV`, and $u(\Sigma^{\texttt{NTRIV}})$ is the reduct of $\Sigma^{\texttt{NTRIV}}$ to an MSA signature). Notice that u is an inclusion since the CafeOBJ cube admits a trivial inclusion system in which all arrows are inclusions, that the reduct from RWL signatures to MSA signatures is an identity, and that $\Sigma^{\texttt{TRIV}} \rightarrow \Sigma^{\texttt{NTRIV}}$ is an inclusion of MSA signatures.

An interesting aspect of this example is given by its model theory. The denotation of this module import is the model reduct functor $\text{MOD}(T^{\texttt{NTRIV}}) \rightarrow \text{MOD}(T^{\texttt{TRIV}})$ in the CafeOBJ institution. From Definition 28, this means $\beta^u_{\Sigma^{\texttt{NTRIV}}}; \text{MOD}^{\text{MSA}}(\varphi)$,

which means a two level reduction. The first level, $\beta^{u}_{\Sigma^{\mathrm{NTRIV}}}$, means getting rid of the transitions of the carrier (i.e. making the carrier discrete) of the model and regarding the interpretation of $\_|\_$ as a function rather than a functor. The second level, $\mathrm{MOD}^{\mathrm{MSA}}(\varphi)$, is a reduction internal to MSA which forgets the interpretation of $\_|\_$. It is very important to notice that the correct denotation for this module import can be achieved only in the framework of the CafeOBJ institution, the fact that this is a Grothendieck institution being crucial. None of the institutions of the CafeOBJ cube (such as RWL for example) would have been appropriate to give the denotation of this example.

We denote the partial order of module imports by $\trianglelefteq$. By following the OBJ tradition, we can distinguish between three basic kinds of imports, **protecting**, **extending**, and **using**. At the level of the language, these should be treated just as semantic declarations which determine the denotation of the importing module from the denotation of the imported module.

Definition 40 *Given a theory morphism $\varphi\colon T \to T'$, and a model M of T, an **expansion of** M **along** φ is a model M' of T' satisfying the following properties:*

- *$M'\restriction_\varphi = M$ iff the expansion is **protecting**,*
- *there is an* injective[20] *model homomorphism $M \hookrightarrow M'\restriction_\varphi$ iff the expansion is **extending**,*
- *there is an arbitrary model homomorphism $M \to M'\restriction_\varphi$ iff the expansion is **using**, and*
- *M' is free over M*[21] *with respect to φ iff the expansion is **free**.*

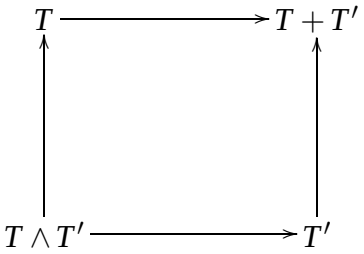
Definition 41 *Fix an import $SP \trianglelefteq SP'$ and let T and T' be the theories corresponding to SP and SP', respectively. Then*

$[\![SP']\!] = \{M' \mid M' \models T',\ M'$ *is an expansion of the same kind as the importation mode involved of some model* $M \in [\![SP]\!]$ *(and in addition free if SP' is initial)* $\}$.

Multiple imports are handled by a lattice structure on imports. The existence of (finite) least upper bounds (called *sums* in [17]) of module imports corresponds to the weak inclusion system of theory morphisms being a proper inclusion system. In [18] we lift sums from inclusion systems for ordinary theory morphisms to extra theory morphisms. The (finite) greatest lower bounds (called *intersections*) are defined as the pullback of the sums.

[20] Under a suitable concept of 'injectivity'.

[21] Which means that M' is the free object over M with respect to the model reduct functor $\_\restriction_\varphi\colon \mathrm{MOD}(T') \to \mathrm{MOD}(T)$.



In practice, one of the important properties of the sum-intersection square is to be a pushout besides being a pullback square. This result for the inclusion system of extra theory morphisms together with the details of its construction are given in [18]. All these can be easily translated to the conceptual famework of the Grothendieck institutions. [22]

4.2 Parameterisation

Parameterization is an important feature of all module systems of modern specification or programming languages. In CafeOBJ the mathematical concept of parameterised modules is based on *injections* (in the sense of Definition 37) in the category of theories of the CafeOBJ institution:

> **Parameterised specifications** $SP(X :: P)$ are modelled as injective theory morphisms from the theory corresponding to the parameter P to the theory corresponding to the body SP. **Views** are modelled as theory morphisms.

The denotation $[\![SP]\!]$ of the body is determined from the denotation of the parameter accordingly to the parameterisation mode involved as in the case of module imports (Definition 41).

We distinguish two opposite approaches on parameters: a *shared* and a *non-shared* one. In the 'non-shared' approach, the multiple parameters are mutually disjoint (i.e., $\mathrm{Im}(X) \wedge \mathrm{Im}(X') = \emptyset$ for X and X' two different parameters, where $\mathrm{Im}(X)$ means the image of the parameter into the body theory and we denote its intersections, or greatest lower bounds, by $\wedge$) and they are also disjoint from any module imports $T_0 \trianglelefteq T$ (i.e., $\mathrm{Im}(X) \wedge T_0 = \emptyset$). In the 'shared' approach this principle is relaxed to being disjoint *outside common imports*, i.e., $\mathrm{Im}(X) \wedge \mathrm{Im}(X') = \sum_{T_1 \trianglelefteq X} T_1 \wedge \sum_{T_1 \trianglelefteq X'} T_1$ for X and X' two different parameters and $\mathrm{Im}(X) \wedge T_0 = \sum_{T_1 \trianglelefteq X} \wedge T_0$ for all $T_0 \trianglelefteq T$. The 'non-shared' approach has the potentiality of a much more powerful module system, while the 'shared' approach seems to be more convenient to

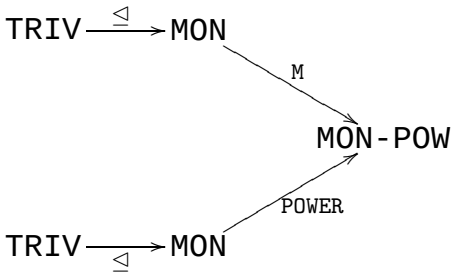
[22] The construction of the inclusion system for Grothendieck institution relies on the construction of finite limits in Grothendieck (fibred) categories.

implement (see [13] for details). The CafeOBJ definition gives the possibility of the whole range of situations between these two extremes by giving the user the possibility to control the sharing.

Example 42 This is an example adapted from [13]. Consider the (double parameterised) specification of a 'power' operation on monoids, where powers are elements of another (abstract) monoid rather than natural numbers.

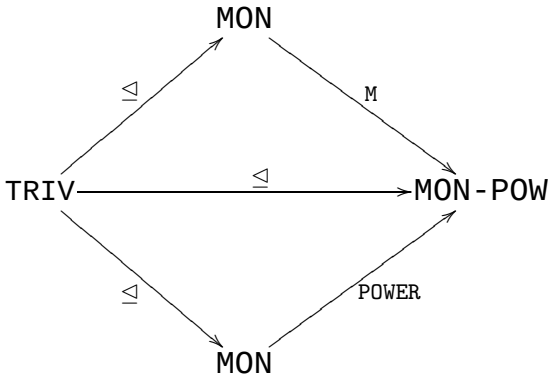
```
mod* MON {
  protecting(TRIV)
  op nil : -> Elt
  op _;_ : Elt Elt -> Elt {assoc id: nil}
}
mod* MON-POW (POWER :: MON, M :: MON)
{
  op _^_ : Elt.M Elt.POWER -> Elt.M
  vars m m' : Elt.M
  vars p p' :  Elt.POWER
  eq (m ; m')^ p   = (m ^ p) ; (m' ^ p) .
  eq  m ^ (p ; p') = (m ^ p) ; (m ^ p') .
  eq  m ^ nil     =  nil .
}
```

The diagram defining MON-POW is



where MON-POW consists of two copies of MON labelled by M and POWER respectively, plus the power operation together with the 3 axioms defining its action. This means TRIV is *not* shared, since the power monoid and the base monoid are allowed to have different carriers. The denotation ⟦MON-POW⟧ consists of all protecting expansions (with interpretations of _^_) to MON-POW of non-shared amalgamations of monoids corresponding to the two parameters.

In the ‘shared’ approach, the parameterisation diagram is



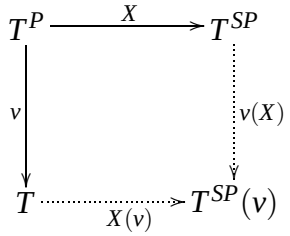
In this case, the denotation $[\![$`MON-POW`$]\!]$ consists of all two different monoid structures on *the same set*, plus an interpretation of `_^_` satisfying the ‘power’ equations.

In CafeOBJ such sharing can be achieved by the user by means of the command **`share`** which has the effect of enforcing that the modules declared as shared are *included* rather than ‘injected’ in the body specification. In this case we have just to specify

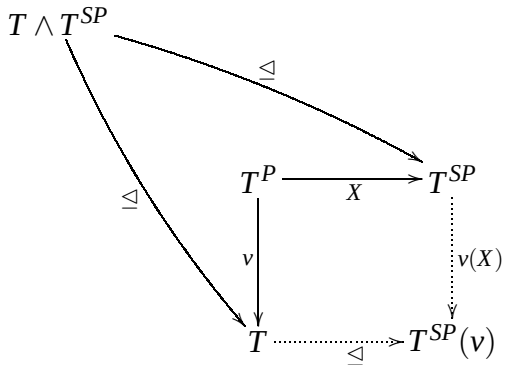
```
share(TRIV)
```

The following defines parameter instantiation by means of the pushout technique for the case of single parameters. This definition can be naturally extended to the case of multiple parameters (for details about instantiation of multiple parameters in CafeOBJ see [13]).

Definition 43 *Let $SP(X :: P)$ be a parameterised module and let $T^P \xrightarrow{X} T^{SP}$ be its representation as theory morphism. Let $v\colon T^P \to T$ be a view. Then the instantiation $T^{SP}(v)$ is given by the following pushout of theory morphisms in the CafeOBJ institution:*



in the 'non-shared' approach, and by the following co-limit

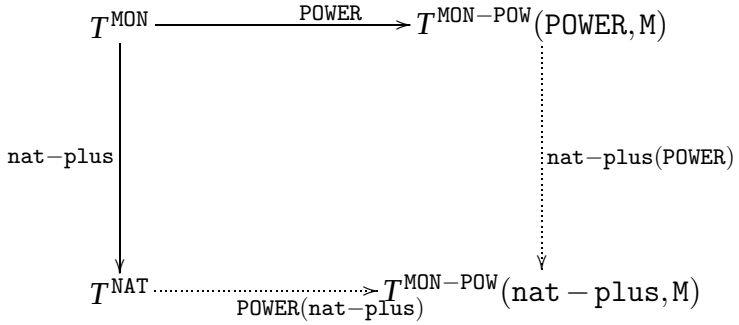


in the 'shared' approach.

Example 44 Consider the following CafeOBJ view interpreting the monoid structure by the natural numbers with addition:

```
view nat-plus from MON to NAT {
  sort Elt -> Nat,
  op _;_ -> _+_,
  op nil -> 0
}
```

We instantiate `MON-POW` by `MON-POW(POWER <= nat-plus, M)` for obtaining monoids with natural powers. The theory pushout corresponding to this instantiation can be represented by:



The semantics of parameter instantiation relies on preservation properties of conservative extensions by pushouts of theory morphisms. Recall the concept of conservative theory morphism from [17]:

Definition 45 *A theory morphism* $\varphi\colon T \to T'$ *is* ***conservative*** *if and only if any model M of T has a protecting expansion along* φ*.*

Example 46 In the case of Example 44, `POWER(nat-plus)` is conservative because the theory morphism `POWER` is conservative. The denotation of `MON-`

`POW(nat-plus,M)` thus consists of all monoids with natural powers.

Preservation of conservative extensions in Grothendieck institutions is a significantly harder problem than in ordinary institutions. Such technical results for Grothendieck institutions have been obtained in [18] but within the conceptual framework of extra theory morphisms.

5 Conclusions and Future Work

We surveyed the logical foundations of CafeOBJ which constitute the origin of the concrete definition of the language [13]. Some of its main features are:

- simplicity and effectiveness via appropriate abstractness,
- cohesiveness,
- flexibility,
- it provides support for multi-paradigm integration,
- it provides support for the development of specification methodologies, and
- it uses state-of-the-art methods in algebraic specification research.

We defined the CafeOBJ institution, overviewed its main properties, and presented the main mathematical concepts and results underlying basic and structured specification in CafeOBJ.

Besides theoretical developments, future work on CafeOBJ will mainly concentrate on specification and verification methodologies, especially the object-oriented ones emerging from the behavioural specification paradigm. This includes refining the existing object composition methodology based on projection operations [33, 15, 13] but also the development of new methodologies and careful identification of the application domains most suitable to certain specification and verification methodologies.

The development of CafeOBJ has been an interplay process among language design, language and system implementation, and methodology development. Although the language design is based on solid and firm mathematical foundations, it has been greatly helped by the existence of a running system, which gave the possibility to run various relevant examples, thus giving important feedback at the level of concrete language constructs and execution commands. The parallel development of methodologies gave special insight on the relationship between the various paradigms co-existing in CafeOBJ with consequences at the level of design of the language constructs.

We think that the interplay among mathematical semantic design of CafeOBJ, the system implementation, and the methodology development has been the most im-

portant feature of CafeOBJ design process. We believe this promises the sound and reasonable development of a practical formal specification method around CafeOBJ.

Acknowledgements

We thank the editors and both anonymous referees for their detailed and careful suggestions and comments which helped improving the presentation of the paper.

References

[1] Rod Burstall and Joseph Goguen. The semantics of Clear, a specification language. In Dines Bjorner, editor, *Proceedings, 1979 Copenhagen Winter School on Abstract Software Specification*, pages 292–332. Springer, 1980. Lecture Notes in Computer Science, Volume 86; based on unpublished notes handed out at the Symposium on Algebra and Applications, Stefan Banach Center, Warsaw, Poland, 1978.

[2] Samuel Buss and Grigore Roşu. Incompleteness of behavioural logics. In Horst Reichel, editor, *Coalgebraic Methods in Computer Science*, volume 33 of *Electronic Notes in Theoretical Computer Science*, pages 61–79. Elsevier Science, 2000.

[3] Manuel Clavel, Steve Eker, Patrick Lincoln, and Jose Meseguer. Principles of Maude. *Electronic Notes in Theoretical Computer Science*, 4, 1996. Proceedings, First International Workshop on Rewriting Logic and its Applications. Asilomar, California, September 1996.

[4] Virgil Emil Căzănescu and Grigore Roşu. Weak inclusion systems. *Mathematical Structures in Computer Science*, 7(2):195–206, 1997.

[5] Răzvan Diaconescu. Category-based semantics for equational and constraint logic programming, 1994. DPhil thesis, University of Oxford.

[6] Răzvan Diaconescu. Completeness of category-based equational deduction. *Mathematical Structures in Computer Science*, 5(1):9–41, 1995.

[7] Răzvan Diaconescu. Behavioural rewriting logic: semantic foundations and proof theory, October 1996. Submitted to publication.

[8] Răzvan Diaconescu. Completeness of semantic paramodulation: a category-based approach. Technical Report IS-RR-96-0006S, Japan Advanced Institute for Science and Technology, 1996.

[9] Răzvan Diaconescu. Foundations of behavioural specification in rewriting logic. *Electronic Notes in Theoretical Computer Science*, 4, 1996. Proceedings, First International Workshop on Rewriting Logic and its Applications. Asilomar, California, September 1996.

[10] Răzvan Diaconescu. Extra theory morphisms for institutions: logical semantics for multi-paradigm languages. *J. of Applied Categorical Structures*, 6(4):427–453, 1998. A preliminary version appeared as JAIST Technical Report IS-RR-97-0032F in 1997.

[11] Răzvan Diaconescu. Category-based constraint logic. *J. Mathematical Structures in Computer Science*, 10(3):373–407, 2000.

[12] Răzvan Diaconescu. Grothendieck institutions. IMAR Preprint 2-2000, Institute of Mathematics of the Romanian Academy, February 2000. ISSN 250-3638.

[13] Răzvan Diaconescu and Kokichi Futatsugi. *CafeOBJ Report: The Language, Proof Techniques, and Methodologies for Object-Oriented Algebraic Specification*, volume 6 of *AMAST Series in Computing*. World Scientific, 1998.

[14] Răzvan Diaconescu and Kokichi Futatsugi. Behavioural coherence in object-oriented algebraic specification. *J. Universal Computer Science*, 6(1):74–96, 2000. First version appeared as JAIST Technical Report IS-RR-98-0017F, June 1998.

[15] Răzvan Diaconescu, Kokichi Futatsugi, and Shusaku Iida. Component-based algebraic specification and verification in CafeOBJ. In Jeannette M. Wing, Jim Woodcock, and Jim Davies, editors, *FM'99 – Formal Methods*, volume 1709 of *Lecture Notes in Computer Science*, pages 1644–1663. Springer, 1999.

[16] Răzvan Diaconescu, Kokichi Futatsugi, and Shusaku Iida. CafeOBJ jewels. In Kokichi Futatsugi, Ataru Nakagawa, and Tetsuo Tamai, editors, *Cafe: An Industrial-Strength Algebraic Formal Method*. Elsevier, 2000.

[17] Răzvan Diaconescu, Joseph Goguen, and Petros Stefaneas. Logical support for modularisation. In Gerard Huet and Gordon Plotkin, editors, *Logical Environments*, pages 83–130. Cambridge, 1993. Proceedings of a Workshop held in Edinburgh, Scotland, May 1991.

[18] Răzvan Diaconescu and Petros Stefaneas. Categorical foundations of modularization for multi-paradigm languages. Technical Report IS-RR-98-0014F, Japan Advanced Institute for Science and Technology, 1998.

[19] Kokichi Futatsugi, Joseph Goguen, Jean-Pierre Jouannaud, and Jose Meseguer. Principles of OBJ2. In *Proceedings of the 12th ACM Symposium on Principles of Programming Languages*, pages 52–66. ACM, 1985.

[20] Joseph Goguen. *Theorem Proving and Algebra*. MIT, 2002. To appear.

[21] Joseph Goguen and Rod Burstall. Institutions: Abstract model theory for specification and programming. *Journal of the Association for Computing Machinery*, 39(1):95–146, January 1992.

[22] Joseph Goguen and Răzvan Diaconescu. An Oxford survey of order sorted algebra. *Mathematical Structures in Computer Science*, 4(4):363–392, 1994.

[23] Joseph Goguen and Răzvan Diaconescu. Towards an algebraic semantics for the object paradigm. In Harmut Ehrig and Fernando Orejas, editors, *Recent Trends in Data Type Specification*, volume 785 of *Lecture Notes in Computer Science*, pages 1–34. Springer, 1994.

[24] Joseph Goguen and Grant Malcolm. A hidden agenda. Technical Report CS97-538, University of California at San Diego, 1997.

[25] Joseph Goguen and José Meseguer. Eqlog: Equality, types, and generic modules for logic programming. In Douglas DeGroot and Gary Lindstrom, editors, *Logic Programming: Functions, Relations and Equations*, pages 295–363. Prentice-Hall, 1986. An earlier version appears in *Journal of Logic Programming*, Volume 1, Number 2, pages 179–210, September 1984.

[26] Joseph Goguen and José Meseguer. Unifying functional, object-oriented and relational programming, with logical semantics. In Bruce Shriver and Peter Wegner, editors, *Research Directions in Object-Oriented Programming*, pages 417–477. MIT, 1987. Preliminary version in *SIGPLAN Notices*, Volume 21, Number 10, pages 153–162, October 1986.

[27] Joseph Goguen and José Meseguer. Order-sorted algebra I: Equational deduction for multiple inheritance, overloading, exceptions and partial operations. *Theoretical Computer Science*, 105(2):217–273, 1992. Also, Programming Research Group Technical Monograph PRG–80, Oxford University, December 1989.

[28] Joseph Goguen, James Thatcher, and Eric Wagner. An initial algebra approach to the specification, correctness and implementation of abstract data types. Technical Report RC 6487, IBM T.J. Watson Research Center, October 1976. In *Current Trends in Programming Methodology, IV*, Raymond Yeh, editor, Prentice-Hall, 1978, pages 80–149.

[29] Joseph Goguen, Timothy Winkler, José Meseguer, Kokichi Futatsugi, and Jean-Pierre Jouannaud. Introducing OBJ. In Joseph Goguen and Grant Malcolm, editors, *Software Engineering with OBJ: algebraic specification in action*. Kluwer, 2000.

[30] Alexandre Grothendieck. Catégories fibrées et descente. In *Revêtements étales et groupe fondamental, Séminaire de Géométrie Algébraique du Bois-Marie 1960/61, Exposé VI*. Institut des Hautes Études Scientifiques, 1963. Reprinted in Lecture Notes in Mathematics, Volume 224, Springer, 1971, pages 145–94.

[31] Rolf Hennicker and Michel Bidoit. Observational logic. In A. M. Haeberer, editor, *Algebraic Methodology and Software Technology*, number 1584 in LNCS, pages 263–277. Springer, 1999. Proc. AMAST'99.

[32] Hendrik Hilberdink. Foundations for rewriting logic. In Kokichi Futatsugi, editor, *The 3rd International Workshop on Rewriting Logic and its Applications*, 2000.

[33] Shusaku Iida, Kokichi Futatsugi, and Răzvan Diaconescu. Component-based algebraic specification: - behavioural specification for component-based software engineering -. In *Behavioral specifications of businesses and systems*, pages 103–119. Kluwer, 1999.

[34] Saunders MacLane. *Categories for the Working Mathematician*. Springer, second edition, 1998.

[35] José Meseguer. Rewriting as a unified model of concurrency. Technical Report SRI-CSL-90-02R, SRI International, Computer Science Laboratory, February 1990. Revised June 1990.

[36] José Meseguer. Conditional rewriting logic as a unified model of concurrency. *Theoretical Computer Science*, 96(1):73–155, 1992.

[37] José Meseguer. Membership algebra as a logical framework for equational specification. In F. Parisi-Pressice, editor, *Proc. WADT'97*, number 1376 in Lecture Notes in Computer Science, pages 18–61. Springer, 1998.

[38] José Meseguer and Joseph Goguen. Initiality, induction and computability. In Maurice Nivat and John Reynolds, editors, *Algebraic Methods in Semantics*, pages 459–541. Cambridge, 1985.

[39] R. Paré and D. Schumacher. *Indexed Categories and their Applications*, volume 661 of *Lecture Notes in Mathematics*, chapter Abstract Families and the Adjoint Functor Theorems, pages 1–125. Springer, 1978.

[40] Andrzej Tarlecki, Rod Burstall, and Joseph Goguen. Some fundamental algebraic tools for the semantics of computation, part 3: Indexed categories. *Theoretical Computer Science*, 91:239–264, 1991. Also, Monograph PRG–77, August 1989, Programming Research Group, Oxford University.