Generic refinements for behavioral specifications

Marius Petria

Doctor of Philosophy

School of Informatics

University of Edinburgh

2010

Abstract

This thesis investigates the properties of generic refinements of behavioral specifications.

At the base of this investigation stands the view from algebraic specification that abstract data types can be modeled as algebras. A specification of a data type is formed from a syntactic part, i.e. a signature detailing the interface of the data type, and a semantic part, i.e. a class of algebras (called its models) that contains the valid implementations of that data type.

Typically, the class of algebras that constitutes the semantics of a specification is defined as the class of algebras that satisfy some given set of axioms. The behavioral aspect of a specification comes from relaxing the requirements imposed by axioms, i.e. by allowing in the semantics of a specification not only the algebras that literally satisfy the given axioms, but also those algebras that appear to behave according to those axioms. Several frameworks have been developed to express the adequate notions of what it means to be a behavioral model of a set of axioms, and our choice as the setting for this thesis will be Bidoit and Hennicker's *Constructor-based Observational Logic*, abbreviated **COL**.

Using specifications that rely on the behavioral aspects defined by **COL** we study the properties of generic refinements between specifications. Refinement is a relation between specifications. The refinement of a target specification by a source specification is given by a function that constructs models of the target specification from the models of the source specification. These functions are called constructions and the source and target specifications that they relate are called the context of the refinement. The theory of refinements between algebraic specifications, with or without the behavioral aspect, has been well studied in the literature. Our analysis starts from those studies and adapts them to **COL**, which is a relatively new framework, and for which refinement has been studied only briefly.

The main part of this thesis is formed by the analysis of generic refinements. Generic refinements are represented by constructions that can be used in various contexts, not just in the context of their definition. These constructions provide the basis for modular refinements, i.e. one can use a locally defined construction in a global context in order to refine just a part of a source specification. The ability to use a refinement outside its original context imposes additional requirements on the construction that represents it. An implementer writing such a construction must not use details of the source models that can be contradicted by potential global context requirements.

This means, roughly speaking, that he must use only the information available in the source signature and also any *a priori* assumption that was made about the contexts of use.

We look at the basic case of generic refinements that are reusable in every global context, and then we treat a couple of variations, i.e. generic refinements for which an *a priori* assumption it is made about the nature of their usage contexts. In each of these cases we follow the same pattern of investigation. First we characterize the constructions that ensure reusability by means of preservation of relations, and then, in most cases, we show that such constructions must be definable in terms of their source signature.

Throughout the thesis we use an informal analogy between generic (i.e. polymorphic) functions that appear in second order lambda calculus and the generic refinements that we are studying. This connection will enable us to describe some properties of generic refinements that correspond to the properties of polymorphic functions inferred from their types and named "theorems for free" by Wadler.

The definability results, the connection between the assumptions made about the usage contexts and the characterizing relations, and the "theorems for free" for behavioral specifications constitute the main contributions of this thesis.

To Irina, for being continuously supportive.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Marius Petria*)

Acknowledgements

I want to thank my supervisor Don Sannella for his valuable guidance during all these years. Not only for the technical expertise but also for the understanding he showed with regard to my particular concerns. His constant support made it possible for this thesis to reach its final form.

I also want to thank the examiners of this thesis Andrzej Tarlecki and John Longley for giving me important feedback that helped improving the quality of the final version. A special thanks to Philip Wadler whose paper about “theorems for free” inspired the work from this thesis.

This work was funded for three years by a Wolfson Microelectronics Scholarship and an ORSAS award.

Last but not the least I want to thank my family and friends for being there when my feeling was that nothing worth reading will come out of my research. Their encouragement pushed me forward and allowed me to continue.

Table of Contents

Chapter 1

Introduction

Contents

1.1 General remarks

Algebraic specification is a paradigm that encompasses powerful techniques for representing systems as algebras and their constraints as axioms. The generality of this framework makes it applicable in all fields where one needs to confirm the soundness of an implemented system with respect to some logical requirements. An important area of usability for algebraic specifications is software development. In this field one could easily arrive at the conclusion that good practices developed only by empirical means are not always enough to guarantee the proper evolution and development of correct software, and hence the need for deductive reasoning based on a formal framework arises.

Firstly, the practice of algebraic specifications teaches one how to represent information. In the case of data that is typically done by using *abstract data types*. Abstract data types are formed from a syntactic part representing the names of the operations that are meaningful for that data type, and from a semantic part representing the class of algebras that are considered as acceptable implementations of that data type. One way to define the class of acceptable algebras in an abstract data type's semantics is to use the tools of universal algebra and logic, by specifying a set of axioms that needs to be

satisfied by those algebras. This track is particularly useful if we want a short presentation of the class of acceptable implementations for a data type, instead of explicitly enumerating the potential algebras. By giving a set of axioms that they should obey we use just the language of a particular logic and the symbols present in the signature. For example we could easily think of natural numbers as bytes of memory in a computer, hence imagining an ad hoc implementation for the data type of natural numbers. But it will be too much to actually define the configuration of each particular computer that can represent numbers in the process of formally presenting the semantics of natural numbers. So, instead of defining the semantics of a data type by explicitly describing the acceptable implementations it is simpler to give a more succinct definition by means of axioms. An important consequence of this methodology is that the choice for simplicity brings a great amount of generality, i.e. one has the freedom to change or to add any details to an implementation as long as it does not violates the requirements of the axioms.

A distinct case of representing information arises in the task of representing programs. This appears in the literature of algebraic specifications as the theory of refinements. A program (or a construction, in the terminology we use later in this thesis) is typically represented by a function between abstract data types, i.e. it takes algebras from its input data type and creates algebras in its output data type. As in the case of actual implementations of data types, definitions of actual programs can be extremely complex, as they might include the entire definition of the language in which they are written. One way in which classes of programs can be tuned to reflect our requirements is by enforcing them on their source and target models, i.e. by adding axioms to the corresponding source and target data types. However, in this thesis we will look at a supplementary method for coercing programs. We will look at programs that are generic, i.e. that are reusable in contexts different than that of their definition. It is important to understand that genericity of constructions is not expressed by axioms of their source or their target specifications but by the ability to reuse them.

The second important practice of algebraic specifications is the use of formal deduction. One can infer "new" properties about an abstract data type by using the defining axioms under the rules of a chosen logical system. By proving a property formally one can ensure that any implementation provided for that data type will have the entailed property. With the combination of these two features, specification and deduction, algebraic specification provides a framework for validating software systems.

Several changes were made to the basic framework of algebraic specifications we

have described, in order to make it more fit for practical specification scenarios. One thing under scrutiny has been the way in which axioms should describe the class of acceptable algebras of a specification. It was found that the standard satisfaction relation, that requires the equality symbol from the axioms to be interpreted as the identity in the carriers of the algebras, is over-restrictive. A more permissive alternative is that the equality symbol from the axioms should be interpreted as *observational equality* rather than as identity. In consequence not only algebras that satisfy the axioms in the classical sense are deemed as acceptable but also those for which no observation which contradicts the axioms can be made. These modifications were included in the design of *behavioral logics* [GM00, BH06a] and guided the development of refinement theory using behavioral semantics [ST87, BH98, EK99].

1.2 The goal of the thesis

In this thesis we look at the properties of generic programs. Generic programs correspond to those constructions (functions between data types) that can be used not only in the context of their definition, but can be reused also in contexts that introduce new details. Technically, genericity can be characterized in terms of preservation of relations (the term *stability* is used to denote preservation of relations between algebras in [Sch87, BST08, STar]). So, without being too formal we can say that reusable (i.e. generic) constructions are those that preserve the relations between their arguments. In addition to this result we develop another method for characterizing generic programs inspired by the work on second order lambda calculus (also known as System F) done in [Gir72, Rey74, GLT89, Wad89]. More explicitly, generic programs will be characterized by definability results. To summarize the result we can say that in order for a program to be reusable it must be written using only the syntax provided by its input data type.

The insight that second order lambda calculus phenomena can be replicated in the theory of algebraic specification is based essentially on the identification of stability (introduced in [Sch87] for algebraic specification) with parametricity (introduced in [Rey74] for System F). Both stability and parametricity were invented to capture the phenomenon of abstraction barriers: stability in the framework of algebraic specifications; and parametricity in second order lambda calculus. We do not attempt to formalize the relation between these two concepts but rather use the apparent similarities to guide our exploration in the algebraic specifications world so as to mimic

older results obtained using parametricity in second order lambda calculus. Something similar to the "theorems for free", noticed by Wadler in [Wad89] for System F, will be obtained using this path for generic programs in algebraic specifications.

The novelty of our research is given partly by the analysis of some older results about stability in a newer framework for behavioral specifications, i.e. Bidoit and Hennicker's *Constructor-based Observational Logic*[BH06a]. The richness of this framework allows us to examine details that have not been expressed in the literature, like the correspondence between the class of potential usage contexts and conditions on the relations characterizing generic programs. Additionally, we examine definability results for generic programs, and this investigation is to our knowledge new in the algebraic specification world. Based on definability we can prove interesting properties of data types that arise from generic constructions. We will name these properties "theorems for free" to emphasize the parallel with the results for second order lambda calculus. The reason why some "theorems" for generically constructed models can be considered as coming "for free" is that they are not a consequence of the constraints imposed by the source and target specification but they are implied by genericity.

1.3 The structure of the thesis

The thesis has six chapters and in the middle four chapters we can find original developments.

Chapter 2 is largely devoted to introducing the notions needed in the later chapters. It recalls the definitions of basic algebraic specifications, of higher order type hierarchies and, towards the end of the introductory part, the notions of behavioral specifications in **COL** and refinement. In this chapter we have placed the first original piece of work of the thesis, which is a discussion about fundamentals of behavioral semantics and is not deeply connected with the core body of the thesis on generic constructions (Chapters 3 to 5). The main point of that section is to analyze the concept of behavioral consistency and to explain why the customary approach to this issue lead to some incomplete results in the literature.

Chapter 3 is dedicated to generic programs, or using the formal terminology introduced in that chapter, to *global* **COL**-*constructions*. We analyze those constructions that can be used in any arbitrary global context and we prove that they are characterized by preservation of relations (see Theorem 3.9). Furthermore, we prove that they must be definable (see Proposition 3.11) but we cannot obtain a complete characterization

via definability (see Example 3.15). At the end of the chapter we illustrate how genericity of constructions can be used in order to derive useful properties about constructed models (see Section 3.5) by obtaining "theorems for free" in the style of those proved by Wadler in [Wad89] as a result of parametricity for polymorphic functions.

Chapter 4 explores some variations of the notion presented in the previous chapter. While in Chapter 3 we deal with *general* global **COL**-constructions, i.e. constructions that are reusable in *any* context, in Chapter 4 we look at constructions that are reusable in a limited number of contexts, typically chosen to work well with the behavioral constraints imposed by **COL**. Thus we examine two cases, the so-called *vertical* global constructions and the *quasi-vertical* ones. For both of them we follow the same pattern of investigation. First we characterize them by means of preservation of relations, then we look at definability properties, and in the end we show how all these features can enhance our knowledge about constructed models by deriving "theorems for free".

Chapter 5 contains another variation of the notion of global construction. We consider signatures that are enhanced with an explicit type structure. Our goal for that chapter is to get closer to the capabilities of second order lambda calculus, in order to make the correspondence between stability and parametricity more accurate (the original "theorems for free" of Wadler were proved for higher order functions). Among things we define the notion of *logical* **COL**-*signature* and in such signatures we prove that a long standing problem, from both the lambda calculus world and the algebraic world, can be solved satisfactorily. More precisely, in Section 5.5, we prove that observational equivalence between algebras can be soundly and completely characterized by the existence of a logical relation.

Chapter 6 concludes with a summary and a comment on possible future directions among which the view that constructions correspond to proofs in the style of the Curry-Howard isomorphism [CF58, How80, Gri90] seems the most interesting.

Chapter 2

Preliminaries

Contents

In this chapter we introduce the basic concepts concerning behavioral specifications and refinements between them. In the beginning of the chapter we recall the fundamentals of universal algebra and of simple type hierarchies represented algebraically. Then we describe a framework that incorporates the behavioral aspects of a specification and we introduce refinements relative to this framework. In the end, Section 2.5.1, we present an original discussion about the appropriate way to define the behavioral semantics of a specification and the problems that appear in the literature in connection to this issue.

2.1 Categories and institutions

We assume the reader is familiar with basic notions and standard notations from category theory; e.g., see [Mac98] for an introduction to this subject. Here we recall very briefly some of them. By way of notation, $|\mathbb{C}|$ denotes the class of objects of a category $\mathbb{C}$, $\mathbb{C}(A,B)$ the set of arrows with domain A and codomain B, and composition is

denoted by ";" and is written in diagrammatic order. The category of sets (as objects) and functions (as arrows) is denoted by $\mathbb{S}et$, and $\mathbb{CAT}$ is the category of all categories. The opposite of a category $\mathbb{C}$ (obtained by reversing the arrows of $\mathbb{C}$) is denoted $\mathbb{C}^{op}$.

The theory of *institutions* [GB92] is a categorical abstract model theory which formalizes the intuitive notion of logical system, including syntax, semantics, and the satisfaction between them. An *institution* $I = (\mathbb{S}ign^I, \mathsf{Sen}^I, \mathsf{Mod}^I, \models^I)$ consists of

1. a category $\mathbb{S}ign^I$, whose objects are called *signatures*,
2. a functor $\mathsf{Sen}^I{:}\mathbb{S}ign^I \to \mathbb{S}et$, giving for each signature a set whose elements are called *sentences* over that signature,
3. a functor $\mathsf{Mod}^I{:}(\mathbb{S}ign^I)^{op} \to \mathbb{CAT}$ 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 $\models^I_\Sigma \subseteq |\mathsf{Mod}^I(\Sigma)| \times \mathsf{Sen}^I(\Sigma)$ for each $\Sigma \in |\mathbb{S}ign^I|$, called Σ-*satisfaction*,

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

$$M' \models^I_{\Sigma'} \mathsf{Sen}^I(\varphi)(\rho) \text{ iff } \mathsf{Mod}^I(\varphi)(M') \models^I_\Sigma \rho$$

holds for each $M' \in |\mathsf{Mod}^I(\Sigma')|$ and $\rho \in \mathsf{Sen}^I(\Sigma)$. We denote the *reduct* functor $\mathsf{Mod}^I(\varphi)$ by $\_{\restriction_\varphi}$ and the sentence translation $\mathsf{Sen}^I(\varphi)$ by $\varphi(\_)$. When $M = M'{\restriction_\varphi}$ we say that M is a φ-*reduct of* M', and that M' is a φ-*expansion of* M. When there is no danger of ambiguity, we may skip the superscripts from the notations of the entities of the institution; for example $\mathbb{S}ign^I$ may be simply denoted $\mathbb{S}ign$.

2.2 Algebraic preliminaries

We will briefly present the basic definitions of many sorted signatures and algebras [EM85]. The concepts will be presented mirroring the constituting parts of an institution: signatures, models, sentences.

An *algebraic signature* Σ is a pair (S, OP) where S is a set of sorts and OP is a set of *operation symbols*. For each operation symbol *op* there is an *operation profile* $s_1, \ldots, s_n \to s$ with $s_1, \ldots, s_n, s \in S$. We write $op : s_1, \ldots, s_n \to s$ to denote an operation with its profile and for simplicity we often write $op : w \to s$ where w stands for the list of argument sorts (also called *arity*) $s_1, \ldots, s_n$. If the list of arguments sorts is

empty we say that the operation symbol is a *constant symbol*. A *signature morphism* $\sigma : \Sigma \to \Sigma'$ between two signatures $\Sigma = (S, OP)$ and $\Sigma' = (S', OP')$ is a pair $(\sigma^{srts}, \sigma^{opns})$ of functions $\sigma^{srts} : S \to S'$ and $\sigma^{opns} : OP \to OP'$ such that for all $op : w \to s \in OP$ we have that $\sigma^{opns}(op) : \sigma^{srts}(w) \to \sigma^{srts}(s) \in OP'$, where $\sigma^{srts}(w) = \sigma^{srts}(s_1) \ldots \sigma^{srts}(s_n)$. We say that a signature morphism is *tight* if it is surjective on sorts.

An *algebra* $A = ((A_s)_{s \in S}, (A_{op})_{op \in OP})$ over a signature (S, OP) consists of an S-sorted family of carriers sets $(A_s)_{s \in S}$ and a family of functions $(A_{op})_{op \in OP}$ such that if $op : s_1, \ldots, s_n \to s$ then A_{op} is a function from $A_{s_1} \times \ldots \times A_{s_n}$ to A_s. The class of algebras corresponding to a signatureΣ is denoted by $\mathsf{Alg}(\Sigma)$.

The reduct of a Σ'-algebra A' w.r.t. a signature morphism $\sigma : \Sigma \to \Sigma'$ is a Σ-algebra A, also denoted by $A'|_\sigma$, such that $A_s = A'_{\sigma(s)}$ for all $s \in S$ and $A_{op} = A'_{\sigma(op)}$ for all $op \in OP$.

The *term algebra* $T_\Sigma(X)$ denotes the free Σ-algebra over variables from X, i.e. its elements are $OP \cup X$-terms. We write T_Σ for the algebra of ground terms $T_\Sigma(\emptyset)$. An *interpretation* $I_\alpha : T_\Sigma(X) \to A$ associated with a *valuation* $\alpha : X \to A$ is defined as usual as the canonical algebra morphism that freely extends the valuation α.

To simplify writing we sometimes denote lists of sorts like $s_1, \ldots, s_n$ by w and hence the profiles of the operations can be written as $op : w \to s$. We also use the notation $\underline{a} : w$ when we want to refer to a tuple of elements $a_1, \ldots, a_n \in A_w = A_{s_1} \times \ldots \times A_{s_n}$. Moreover, sometimes we will omit to explicitly name the sort of the carriers, understanding implicitly that we work in a many-sorted framework even if the notation does not show it (for example we will write $a \in A$ to denote an element in a carrier set of the algebra A and so on). The value of a term t with variables from X under a specific valuation $\alpha : X \to A$ will be written $A_t[\alpha]$ instead of $I_\alpha(t)$.

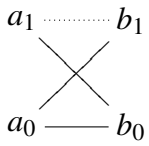
In order to fit the definition of an institution we should also define the sentences that are used to express properties about algebras. These sentences are the usual first order formulas built from equational atoms (written as $t_0 = t_1$) by iterative application of logical connectives and quantifiers. The satisfaction of sentences by models is the usual satisfaction relation defined inductively on the structure of the sentences. The obtained institution is *many sorted first order logic with equality*, abbreviated **FOL**. We will treat **FOL** as the *standard* logic for the rest of the thesis, and we will refer to its models as *standard algebras* when we want to emphasize the difference between them and other kind of algebras, for example behavioral algebras as introduced below. We will also use the term *literal satisfaction* to refer to **FOL** satisfaction when we need to distinguish it from behavioral satisfaction.

2.2.1 Relations

We will now give the definition of a closed algebraic relation. The closedness principle will become important in the next chapters as it intuitively allows the representation of "equivalence relation between different algebras". This concept is taken from the work Sannella and Tarlecki did on behavioral refinements in [STar].

Definition 2.1 (Closed algebraic relation)**.** *Let $\Sigma = (S, OP)$ be an algebraic signature and let A and B be two Σ-algebras. An* algebraic relation*, written $\rho\colon A \leftrightarrow B$, is an S-sorted binary relation $\rho = (\rho_s)_{s\in S}$ that commutes with the operations from OP, i.e. for all $op : s_1, \ldots, s_n \to s \in OP$, for all $a_i \in A_{s_i}$ and $b_i \in B_{s_i}$ such that $a_i\ \rho_{s_i}\ b_i$ for all $i = 1, \ldots, n$ we have that $A_{op}(a_1, \ldots, a_n)\ \rho_s\ B_{op}(b_1, \ldots, b_n)$.*

An algebraic relation $\rho\colon A \leftrightarrow B$ is closed *if for all $a_0, a_1 \in A$ and $b_0, b_1 \in B$ such that $a_0\ \rho\ b_0$, $a_0\ \rho\ b_1$ and $a_1\ \rho\ b_0$ we have that $a_1\ \rho\ b_1$.*



Before going any further we will spell out some useful definitions for manipulating relations. We say that $\rho_0\colon A \leftrightarrow B$ is *finer* than $\rho_1\colon A \leftrightarrow B$ if $\rho_0 \subseteq \rho_1$; we can also say that ρ_1 is *coarser* than ρ_0. We say that a relation $\rho\colon A \leftrightarrow B$ is *bi-surjective* if for all $a \in A$ there exists $b \in B$ such that $a\ \rho\ b$ and vice versa. We say that a relation $\rho\colon A \leftrightarrow B$ is *bi-injective* if for all $a \in A$ and $b_0, b_1 \in B$ such that $a\ \rho\ b_0$ and $a\ \rho\ b_1$ we have that $b_0 = b_1$, and vice versa. We denote by $dom(rel)$ the domain of a relation ρ between A and B. For an algebraic relation $\rho\colon A \leftrightarrow B$ we denote by $A^{dom(\rho)}$, and resp. $B^{dom(\rho)}$, the subalgebras obtained after restricting A, and resp. B, to the domain of ρ.

2.2.2 Higher order types

We will now present the details of considering an explicit type structure in algebraic signatures in a style similar to the one used by Meinke in [Mei92] (earlier definitions in the same style were given in [MTW87]) . These definitions are first used in Chapter 5, because results until then do not need higher order types represented in algebraic signatures.

Definition 2.2. *Let B be a non-empty set, the members of which will be termed* basic types*, the set B being termed a* type basis*. The* simple type hierarchy *$Types(B)$ gener-*

ated by B is defined as $Types(B) = \bigcup_n Types_n(B)$ *where* $Types_n(B)$ *is defined inductively as:*

- $Types_0(B) = B$
- $Types_{n+1}(B) = Types_n(B) \cup \{(s_0 \Rightarrow s_1) \mid s_0, s_1 \in Types_n(B)\}$

Each element $(s_0 \Rightarrow s_1) \in Types(B)$ *is termed a* function type *or a* higher order type. *We use the typical convention of right-associativity for* $\Rightarrow$*, i.e.* $s_0 \Rightarrow s_1 \Rightarrow s_2$ *denotes* $s_0 \Rightarrow (s_1 \Rightarrow s_2)$.

A type structure *S over a type basis B is a subset* $S \subseteq Types(B)$ *which is closed under inner types, i.e.* $(s_0 \Rightarrow s_1) \in S$ *implies* $s_0, s_1 \in S$.

Definition 2.3 (Simply typed signature)**.** *A signature* $\Sigma = (S, OP)$ *is a* simply typed signature *if S is a type structure over some type basis B and for each function type* $(s_0 \Rightarrow s_1)$ *there exists in OP a binary* application operation symbol*:* $app_{s_0,s_1} : s_0 \Rightarrow s_1, s_0 \to s_1$. *We write* APP_S *for the set of all application operation symbols corresponding to the types in S.*

Definition 2.4 (Combinators)**.** *Let S be a type structure. For any* $s_0, s_1, s_2 \in S$ *such that* $(s_0 \Rightarrow s_1 \Rightarrow s_2) \Rightarrow (s_0 \Rightarrow s_1) \Rightarrow s_0 \Rightarrow s_2 \in S$ *we denote by* $\mathcal{S}_{s_0,s_1,s_2}$ *a constant operation on the sort* $(s_0 \Rightarrow s_1 \Rightarrow s_2) \Rightarrow (s_0 \Rightarrow s_1) \Rightarrow s_0 \Rightarrow s_2$. *For any* $s_0, s_1 \in S$ *such that* $s_0 \Rightarrow s_1 \Rightarrow s_0 \in S$ *we denote by* $\mathcal{K}_{s_0,s_1}$ *a constant operation on the sort* $s_0 \Rightarrow s_1 \Rightarrow s_0$. *We denote by* $COMB_S$ *the set of combinators on the sorts from S, i.e.*

$$COMB_S = \{\mathcal{S}_{s_0,s_1,s_2} \mid s_0, s_1, s_2 \in S\} \cup \{\mathcal{K}_{s_0,s_1} \mid s_0, s_1 \in S\}$$

Please note that for a combinator to appear in the set $COMB_S$ it is necessary that the corresponding type is present in S.

Definition 2.5 (Combinatorial signatures)**.** *A* combinatorial signature *is a simply typed signature* $\Sigma = (S, OP)$ *that has all the corresponding combinators, i.e.* $COMB_S \subseteq OP$.

We will frequently use the lambda notation for terms written in a combinatorial signature. It is well known [CF58] that the availability of $\mathcal{S}$ and $\mathcal{K}$ in an algebraic signature is sufficient to give meaning to expressions that use bound variables. Hence we will use lambda terms like $\lambda x{:}s.\lambda f{:}s \Rightarrow s.f(x)$ to denote an $\mathcal{S}, \mathcal{K}$-term of sort $s \Rightarrow (s \Rightarrow s) \Rightarrow s$ under the classical encoding. We will not go into the details of formally introducing lambda terms and their encodings into combinatorial terms and we will defer the reader interested in such introduction to Barendregt's introduction to lambda calculus [Bar81].

Definition 2.6 (Simply typed signature morphisms)**.** *Let* Σ *and* Σ' *be two simply typed signatures. A* simply typed signature morphism σ *between* Σ *and* Σ' *is a standard signature morphism that respects the type structure, i.e.* $\sigma(s_0 \Rightarrow s_1) = \sigma(s_0) \Rightarrow \sigma(s_1)$ *and maps application symbols to application symbols. If in addition* Σ *and* Σ' *are combinatorial, a simply typed signature morphism is* combinatorial *if it also maps combinators to combinators.*

Definition 2.7 (Extensionality axioms)**.** *Let* Σ *be a simply typed signature. We denote by* $ExtAx_\Sigma$ *the set of extensionality axioms* $ExtAx_{s_0,s_1}$ *for all sorts* $s_0 \Rightarrow s_1 \in S$*, where*

$ExtAx_{s_0,s_1}$ *is defined as* $\forall f,g{:}s_0 \Rightarrow s_1.((\forall x{:}s_0.app_{s_0,s_1}(f,x) = app_{s_0,s_1}(g,x)) \longrightarrow f = g)$

Definition 2.8 (Combinatorial axioms)**.** *Let* Σ *be a combinatorial signature. We denote by* $CombAx_\Sigma$ *the set of combinatorial axioms* $CombAx^{\mathcal{S}}_{s_0,s_1,s_2}$ *and* $CombAx^{\mathcal{K}}_{s_0,s_1}$ *for all sorts* $s_0, s_1, s_2 \in S$*, where*

$$CombAx^{\mathcal{K}}_{s_0,s_1} \text{ is defined as } \forall x : s_0, y : s_1 . \mathcal{K}_{s_0,s_1}(x)(y) = x$$

and

$CombAx^{\mathcal{S}}_{s_0,s_1,s_2}$ *is defined as* $\forall f : s_0 \Rightarrow s_1 \Rightarrow s_2, g : s_0 \Rightarrow s_1, x : s_0 . \mathcal{S}_{s_0,s_1,s_2}(f)(g)(x) = (fx)(gx)$

We can now define the notion of *logical relation* which incorporates the extensionality requirement for relations.

Definition 2.9 (Logical relation)**.** *Let* $\Sigma = (S, OP)$ *be a simply typed signature. An algebraic relation* $\rho{:}\, A \leftrightarrow B$ *is* logical *if for all* $s_0 \Rightarrow s_1 \in S$*,* $f \in A_{s_0 \Rightarrow s_1}$ *and* $g \in B_{s_0 \Rightarrow s_1}$ *that are extensionally equal w.r.t* ρ*, i.e.* $f(a)\ \rho_{s_1}\ g(b)$ *for all* $a\ \rho_{s_0}\ b$*, we have that* $f\ \rho_{s_0 \Rightarrow s_1}\ g$*.*

2.3 Constructor-based observational logic

Constructor-based observational logic was introduced in [BH06a] by Bidoit and Hennicker, with the purpose of representing in a simple manner the concepts of reachability and of observability at the signature level. Reachability and observability are essentially concepts expressible in second order logic, but by including them in the definition of signatures one allows rich specifications of data types using only first order formulas. The reachability constraint is enforced by means of constructors which intuitively identify those elements in the carriers of algebras which are important to the user. The

observability constraint is based on observers which induce an observational equality between elements representing the extent to which a user of the algebra can distinguish between elements.

Definition 2.10 (**COL**-signature)**.** *A constructor is an operation symbol* $cons : s_1, \ldots, s_n \to s$ *with* $n \geq 0$*. The result sort* s *of cons is called a* constrained sort *(or* constructed sort*). An* observer *is a pair* (obs, i) *where* $obs : s_1, \ldots, s_i, \ldots, s_n \to s$ *is an operation symbol with* $1 \leq i \leq n$*. The distinguished argument sort* s_i *of obs is called a* state sort (or hidden sort)*.*

A **COL**-signature $\Sigma_{\mathbf{COL}}$ *consists of a signature* $\Sigma = (S, OP)$*, a set* $OP_{Cons} \subseteq OP$ *of constructors and a set* OP_{Obs} *of observers* (obs, i) *with* $obs \in OP$*.*

The set $S_{Cons} \subseteq S$ *of* constrained sorts *(w.r.t.* OP_{Cons}*) consists of all sorts* s *such that there exists at least one constructor in* OP_{Cons} *with range* s*. The set* $S_{Loose} \subseteq S$ *of* loose sorts *consists of all sorts which are not constrained, i.e.* $S_{Loose} = S \setminus S_{Cons}$*.*

The set $S_{State} \subseteq S$ *of* state sorts *(or* hidden sorts*), w.r.t.* OP_{Obs}*, consists of all sorts* s_i *such that there exists at least one observer* $(obs, i) \in OP_{Obs}$*,* $obs : s_1, \ldots, s_i, \ldots, s_n \to s$*. The set* $S_{Obs} \subseteq S$ *of* observable sorts *(or* visible sorts*) consists of all sorts which are not state sorts, i.e.* $S_{Obs} = S \setminus S_{State}$*.*

An observer (obs, i)*, where* $obs : s_1, \ldots, s_i, \ldots, s_n \to s$*, is called a* direct observer *if* $s \in S_{Obs}$*, otherwise is called an* indirect observer*.*

In the following, whenever we have a **COL**-signature $\Sigma_{\mathbf{COL}}$ we use Σ to refer to the underlying algebraic signature, i.e. $\Sigma_{\mathbf{COL}} = (\Sigma, OP_{Cons}, OP_{Obs})$, and similarly for $\Sigma'_{\mathbf{COL}}$.

Remark 2.11. *In [BH06a] Bidoit and Hennicker make the assumption that the nature of sorts is inferable directly from the choice of constructors and observers as in the above definitions. In general we will use the same approach but there are some edge cases when we will need a slightly different setting in which we are allowed to mark explicitly sorts as being constructed or hidden. Basically we want the ability to mark a sort as being constructed even if we have no constructor for that sort, or to mark a sort as being hidden even if we have no observer for it. In the first case our intention is to specify an empty set of constructed elements (because there is no constructor to construct them); while in the second case we would like to specify an observational equality under which all elements appear to be equal (because there is no observer to distinguish between them).*

Therefore, we will extend the collection of **COL** *signatures with signatures written as* $(\Sigma, OP_{Cons}, OP_{Obs}, S_{State}, S_{Cons})$*, where the set of constructed and hidden sorts,* S_{Cons} *and* S_{State}*, are given explicitly. However, we require that this extension is conservative, i.e. that each sort for which there exists a constructor or an observer is included in the corresponding set. In other words we do not change the nature of constructed or hidden sorts that is inferred from the presence of constructors or observers, but we can specify constraints for those sorts that are not already constrained by operations. For example the following signature can be written in the extended setting*

$$(S = \{s_0, s_1\}, OP = \emptyset, OP_{Cons} = \emptyset, OP_{Obs} = \emptyset, S_{State} = \{s_0\}, S_{Cons} = \{s_1\})$$

in order to express that s_0 *is a hidden sort even if it has no observer on it and that* s_1 *is a constrained sort even if it has no constructor. The intended semantics for these limit cases is that the inhabitants of sort* s_0 *are not distinguishable one from another and respectively that there is no inhabitant of sort* s_1*.*

By abusing the notation we will call the extended framework also **COL** *for two reasons. The first reason is that we believe that these limit cases rightfully belong to a framework that wishes to capture observability and reachability constraints completely. The second reason for abusing the original name is that the introduction of these kind of signatures are not fundamental changes of the original setting presented in [BH06a]. The desired effect, i.e. empty sorts and totally indistinguishable elements, are features that can easily be specified by axioms in the original setting. However we will appeal to these extensions just when they are absolutely necessary (for example in Proposition 3.8) and we will do the rest of the presentation in terms of the original setting.*

In order to define the set of important elements in an algebra w.r.t. a **COL**-signature we define the notion of *constructor term*, which are terms built with the aid of distinguished constructors over variables of loose sort.

Definition 2.12 (Constructor term)**.** *Let* $\Sigma_{\mathbf{COL}}$ *be a* **COL***-signature, and let* $X = (X_s)_{s \in S_{Loose}}$ *be a family of countably infinite sets* X_s *of variables of loose sort s. For all* $s \in S_{Cons}$ *, the set* $\mathcal{T}(\Sigma_{\mathbf{COL}})_s$ *of constructor terms with "constrained result sort" s is inductively defined as follows:*

- *Each constant cons* $:\rightarrow s \in OP_{Cons}$ *belongs to* $\mathcal{T}(\Sigma_{\mathbf{COL}})_s$*.*
- *For each constructor cons* $: s_1, \ldots, s_n \rightarrow s \in OP_{Cons}$ *with* $n \geq 1$ *and terms* $t_1, \ldots, t_n$

such that t_i is a variable $x_i : s_i$ if $s_i \in S_{Loose}$ and $t_i \in \mathcal{T}(\Sigma_{\mathbf{COL}})_{s_i}$ if $s_i \in S_{Cons}$,
$cons(t_1, \ldots, t_n) \in \mathcal{T}(\Sigma_{\mathbf{COL}})_s$.

The set of all constructor terms is denoted by $\mathcal{T}(\Sigma_{\mathbf{COL}})$.

Please notice that we cannot guarantee that for every constrained sort there is at least one constructor term of that result sort.

The set of elements generated by interpreting the corresponding loose variables in all the constructor terms represents the **COL**-generated part of an algebra, i.e. the important elements w.r.t. the distinguished constructors.

Definition 2.13 ($\Sigma_{\mathbf{COL}}$-generated part)**.** *Let $\Sigma_{\mathbf{COL}}$ be a* **COL***-signature. For any Σ-algebra $A \in \mathsf{Alg}(\Sigma)$, the* $\Sigma_{\mathbf{COL}}$-generated part *of A is an S-sorted family of sets $Gen_{\Sigma_{\mathbf{COL}}}(A) = (Gen_{\Sigma_{\mathbf{COL}}}(A)_s)_{s \in S}$ defined as follows.*

- *Case $s \in S_{Loose}$: $Gen_{\Sigma_{\mathbf{COL}}}(A)_s = A_s$*
- *Case $s \in S_{Cons}$: $Gen_{\Sigma_{\mathbf{COL}}}(A)_s = \{a \in A_s \mid$ there exists a term $t \in \mathcal{T}(\Sigma_{\mathbf{COL}})_s$ and a valuation $\alpha : X \to A$ such that $A_t[\alpha] = a\}$*

The $\Sigma_{\mathbf{COL}}$-generated algebra *of A denoted by $\langle Gen_{\Sigma_{\mathbf{COL}}}(A)\rangle$ is the minimal subalgebra of A that contains $Gen_{\Sigma_{\mathbf{COL}}}(A)$.*

Let B be a subset of A. We write $Gen^B_{\Sigma_{\mathbf{COL}}}(A)$ for the minimal set generated with the rules presented above and in addition that contains all elements from B. Accordingly, $\langle Gen^B_{\Sigma_{\mathbf{COL}}}(A)\rangle$ is the minimal subalgebra of A that contains $Gen^B_{\Sigma_{\mathbf{COL}}}(A)$.

Please notice that for constrained sorts for which there is no constructor (see Remark 2.11) the generated part on those sorts will be empty.

Dually to constructor terms one can define the possible observations as terms of visible result sort formed only using the distinguished observers. *Contexts* have a slot for the observed element and potentially a finite number of variables that are placeholders for elements in the generated part.

Definition 2.14 (Observable context)**.** *Let $\Sigma_{\mathbf{COL}}$ be an observational signature, let $X = (X_s)_{s \in S}$ be a family of countably infinite sets X_s of variables of sort s and let $Z = (\{z_s\})_{s \in S_{State}}$ be a disjoint family of singleton sets (one for each state sort). For all $s \in S_{State}$ and $v \in S_{Obs}$, the set $\mathcal{C}(\Sigma_{\mathbf{COL}})_{s \to v}$ of* observable $\Sigma_{\mathbf{COL}}$-contexts *with "application sort" s and "observable result sort" v is defined as follows:*

1. *For each direct observer* (obs, i) *with* $obs : s_1, \ldots, s_i, \ldots, s_n \to v$ *and pairwise different variables* $x_1 : s_1, \ldots, x_n : s_n$ *from* X, $obs(x_1, \ldots, x_{i-1}, z_{s_i}, x_{i+1}, \ldots, x_n) \in C(\Sigma_{\mathbf{COL}})_{s_i \to v}$.

2. *For each observable context* $ctx \in C(\Sigma_{\mathbf{COL}})_{s \to s'}$ *, for each indirect observer* (obs, i) *with* $obs : s_1, \ldots, s_i, \ldots, s_n \to s$, *and pairwise different variables* $x_1 : s_1, \ldots, x_n : s_n$ *from* X *not occurring in ctx,* $ctx[obs(x_1, \ldots, x_{i-1}, z_{s_i}, x_{i+1}, \ldots, x_n)/z_s] \in C(\Sigma_{\mathbf{COL}})_{s_i \to v}$ *where* $ctx[obs(x_1, \ldots, x_{i-1}, z_{s_i}, x_{i+1}, \ldots, x_n)/z_s]$ *denotes the term obtained from ctx by substituting the term* $obs(x_1, \ldots, x_{i-1}, z_{s_i}, x_{i+1}, \ldots, x_n)$ *for* z_s.

The observational equality is a reflexive relation defined on the whole carrier of an algebra which relates the elements that cannot be distinguished by applying observable contexts instantiated with all possible generated elements.

Definition 2.15 (Observational $\Sigma_{\mathbf{COL}}$-equality)**.** *Let* $\Sigma_{\mathbf{COL}}$ *be a* **COL***-signature. For any* Σ*-algebra* $A \in \mathsf{Alg}(\Sigma)$*, the observational* $\Sigma_{\mathbf{COL}}$*-equality on* A *is an* S*-sorted binary relation* $\approx_{\Sigma_{\mathbf{COL}},A} = (\approx_{\Sigma_{\mathbf{COL}},A_s})_{s \in S}$ *defined as follows. For all* $s \in S$, *two elements* $a, b \in A_s$ *are observationally equal w.r.t.* $\Sigma_{\mathbf{COL}}$*, i.e. ,* $a \approx_{\Sigma_{\mathbf{COL}},A_s} b$ *(or, for short,* $a \approx_{\Sigma_{\mathbf{COL}},A} b$*), if and only if*

Case $s \in S_{Obs}$*:* $a = b$

Case $s \in S_{State}$*: for all observable sorts* $v \in S_{Obs}$ *, for all observable contexts* $ctx \in C(\Sigma_{\mathbf{COL}})_{s \to v}$*, and for all valuations* $\alpha : X \to Gen_{\Sigma_{\mathbf{COL}}}(A)$ *we have that* $A_{ctx}[\alpha](a) = A_{ctx}[\alpha](b)$

The **COL** *satisfaction relation* (or *behavioral satisfaction*) between Σ-algebras and first-order Σ-sentences is denoted by $\models_{\Sigma_{\mathbf{COL}}}$ and is defined inductively on the structure of sentences by taking equality to denote observational equality.

Definition 2.16 (**COL**-satisfaction)**.** *Let* $\Sigma_{\mathbf{COL}}$ *be a* **COL***-signature and* A *be a* Σ*-algebra.*

atomic formulas $A \models_{\Sigma_{\mathbf{COL}}} l = r$ *if and only if* $A_l \approx_{\Sigma_{\mathbf{COL}},A} A_r$

universal formulas $A \models_{\Sigma_{\mathbf{COL}}} \forall X.e'$ *with* $e' \in \mathsf{Sen}(\Sigma \cup X)$ *if and only if for all* $A' \in \mathsf{Alg}(\Sigma \cup X)$ *such that* $A'_x \in Gen_{\Sigma_{\mathbf{COL}}}(A)$ *for all* $x \in X$ *we have that* $A' \models_{\Sigma'_{\mathbf{COL}}} e'$*. In particular,* $A \models_{\Sigma_{\mathbf{COL}}} \forall X.l = r$ *if and only if for all valuations* $\alpha : X \to Gen_{\Sigma_{\mathbf{COL}}}(A)$ *we have that* $A_l[\alpha] \approx_{\Sigma_{\mathbf{COL}},A} A_r[\alpha]$

existential formulas $A \models_{\Sigma_{\mathbf{COL}}} \exists X.e'$ *with* $e' \in \mathsf{Sen}(\Sigma \cup X)$ *if and only if there exists* $A' \in \mathsf{Alg}(\Sigma \cup X)$ *such that* $A'_x \in Gen_{\Sigma_{\mathbf{COL}}}(A)$ *for all* $x \in X$ *and* $A' \models_{\Sigma'_{\mathbf{COL}}} e'$. *In particular,* $A \models_{\Sigma_{\mathbf{COL}}} \exists X.l = r$ *if and only if there exists a valuation* $\alpha : X \to Gen_{\Sigma_{\mathbf{COL}}}(A)$ *such that* $A_l[\alpha] \approx_{\Sigma_{\mathbf{COL}},A} A_r[\alpha]$

boolean connectors *the definitions for the usual boolean connectors* $\vee, \wedge, \neg$ *are done recursively in a classical way.*

We will denote by $\mathsf{Alg}(\Sigma_{\mathbf{COL}}, Ax)$ *the class of* Σ*-algebras that* **COL***-satisfy the set of formulas Ax.*

Now, we will define what are the acceptable algebras for a **COL** signature. These will be those algebras that respect the reachability constraints and observability constraints imposed by the **COL** signature.

Definition 2.17 (**COL**-algebra)**.** *Let* $\Sigma_{\mathbf{COL}}$ *be a* **COL***-signature. A* $\Sigma_{\mathbf{COL}}$- algebra *(also called* **COL**-algebra*) is a* Σ*-algebra A which satisfies the following constraints induced by* $\Sigma_{\mathbf{COL}}$.

- reachability constraint*: for any* $a \in \langle Gen_{\Sigma_{\mathbf{COL}}}(A) \rangle$ *there exists* $b \in Gen_{\Sigma_{\mathbf{COL}}}(A)$ *such that* $a \approx_{\Sigma_{\mathbf{COL}},A} b$.
- observability constraint*:* $\approx_{\Sigma_{\mathbf{COL}},A}$ *is a* Σ*-congruence on* $\langle Gen_{\Sigma_{\mathbf{COL}}}(A) \rangle$.

The class of all $\Sigma_{\mathbf{COL}}$*-algebras is denoted by* $\mathsf{Alg}_{\mathbf{COL}}(\Sigma_{\mathbf{COL}})$ *and that of* **COL***-algebras that* **COL***-satisfy a set of axioms Ax is denoted by* $\mathsf{Alg}_{\mathbf{COL}}(\Sigma_{\mathbf{COL}}, Ax)$.

A **COL**-algebra which is equal to its generated subalgebra is called *reachable*. A **COL**-algebra for which the observational equality is equal to identity is called *fully-abstract*.

Example 2.18 (Failures of reachability and observability constraints)**.** *From the definition of* **COL***-algebras we can see that there are two ways in which a standard algebra can fail to comply with the requirements of a* **COL***-signature. Both kinds of failures can be understand better if we concentrate our attention on operations that are neither constructors nor observers, and for the sake of brevity let us name them* plain operations*. It is easy to imagine examples in which plain operations would cause the failure of reachability or observability requirements. Consider the following* **COL** *signature:*

$$\Sigma_{\mathbf{COL}} = (S = \{s, v\}, OP = \{a : s, obs : s \to v, op : s \to s\}, OP_{Cons} = \{a, obs\}, OP_{Obs} = \{obs\})$$

and let A be a Σ-algebra such that $A_s = A_v = \mathbb{B}$, $A_a = true$, $A_{obs} = id_{\mathbb{B}}$ and $A_{op} = \neg_{\mathbb{B}}$, where $\mathbb{B}$ is the set of booleans $\{true, false\}$, $id_{\mathbb{B}}$ is the identity function and $\neg_{\mathbb{B}}$ is the negation function. It is clear that $S_{State} = \{s\}$, $S_{Obs} = \{v\}$, $S_{Cons} = S$ and $S_{Loose} = \emptyset$ and therefore the observational equality $\approx_{\Sigma_{\mathbf{COL}},A}$ coincides with identity and $Gen_{\Sigma_{\mathbf{COL}}}(A)_s = true$. It is now obvious that $A_{op} : A_s \to A_s$ does not respect the reachability constraint as $A_{op}(true) = false$ is not behaviorally equal to a generated element (which can only be true). For a counterexample that deals with observability constraints consider the following signature

$$\Sigma_{\mathbf{COL}} = (S = \{s, v\}, OP = \{obs : s \to v, op : s \to v\}, OP_{Cons} = \emptyset, OP_{Obs} = \{obs\})$$

and let A be a Σ-algebra such that $A_s = A_v = \mathbb{B}$, $A_{obs} = \lambda x.true$, $A_{op} = id_{\mathbb{B}}$. Now we have $S_{State} = \{s\}$, $S_{Obs} = \{v\}$, $S_{Cons} = \emptyset$ and $S_{Loose} = S$ and we can easily see that $\approx_{\Sigma_{\mathbf{COL}},A_s} = (\mathbb{B} \times \mathbb{B})$, however A_{op} does not preserve the observational equality.

The notion of morphism between two **COL**-algebras is different from the notion of homomorphism between standard algebras. The main features of **COL**-homomorphisms are that they are relations instead of functions and that they are required to relate at least the elements from the generated algebras and to be compatible with the observational equalities. Basically, a **COL**-homomorphism is a standard homomorphism between quotient algebras obtained after factoring through observational equality.

Definition 2.19 (**COL**-algebra morphism)**.** *Let $\Sigma_{\mathbf{COL}}$ be a* **COL***-signature and let A, B be two $\Sigma_{\mathbf{COL}}$-algebras. A* **COL**-algebra morphism *h: $A \to B$ is an S-sorted family $(h_s)_{s \in S}$ of relations $h_s \subseteq \langle Gen_{\Sigma_{\mathbf{COL}}}(A)\rangle_s \times \langle Gen_{\Sigma_{\mathbf{COL}}}(B)\rangle_s$ with the following properties for all $s \in S$:*

- *for all $a \in \langle Gen_{\Sigma_{\mathbf{COL}}}(A)\rangle_s$, there exists $b \in \langle Gen_{\Sigma_{\mathbf{COL}}}(B)\rangle_s$ such that $a\ h_s\ b$*
- *for all $a \in \langle Gen_{\Sigma_{\mathbf{COL}}}(A)\rangle_s, b, b' \in \langle Gen_{\Sigma_{\mathbf{COL}}}(B)\rangle_s$, if $a\ h_s\ b$, then ($a\ h_s\ b'$ if and only if $b \approx_{\Sigma_{\mathbf{COL}},B} b'$)*
- *for all $a, a' \in \langle Gen_{\Sigma_{\mathbf{COL}}}(A)\rangle_s, b \in \langle Gen_{\Sigma_{\mathbf{COL}}}(B)\rangle_s$, if $a\ h_s\ b$ and $a \approx_{\Sigma_{\mathbf{COL}},A} a'$, then $a'\ h_s\ b$*
- *for all $op : s_1, \ldots, s_n \to s \in OP$, for all $a_i \in \langle Gen_{\Sigma_{\mathbf{COL}}}(A)\rangle_{s_i}$ and $b_i \in \langle Gen_{\Sigma_{\mathbf{COL}}}(B)\rangle_{s_i}$, if $a_i\ h_{s_i}\ b_i$ for $i = 1, \ldots, n$ then $A_{op}(a_1, \ldots, a_n)\ h_s\ B_{op}(b_1, \ldots, b_n)$.*

For a given **COL**-signature $\Sigma_{\mathbf{COL}}$, the $\Sigma_{\mathbf{COL}}$-algebras together with the $\Sigma_{\mathbf{COL}}$-morphisms form a category. The identity morphism id_A on a $\Sigma_{\mathbf{COL}}$-algebra A is given

by the observational equality $\approx_{\Sigma_{\mathbf{COL}},A}$. We denote by $\equiv_{\Sigma_{\mathbf{COL}}}$ the isomorphism relation between $\Sigma_{\mathbf{COL}}$-algebras, i.e. we write $A \equiv_{\Sigma_{\mathbf{COL}}} B$ whenever there exists a **COL**-isomorphism between A and B. In order to get a better idea about the characteristics of **COL**-isomorphisms one can refer to [BH06a]. Please notice that a **COL**-isomorphism is bi-surjective on loose sorts and bi-injective on visible sorts; this observation will be useful in the chapters that follow where various kinds of relations (named *correspondences*) are presented in order to capture the notion of **COL**-isomorphism.

Definition 2.20 (Black box)**.** *Let $\Sigma_{\mathbf{COL}}$ be a* **COL***-signature and A be a $\Sigma_{\mathbf{COL}}$-algebra. The* black box *of A is a* **COL***-algebra equal to the quotient algebra $\langle Gen_{\Sigma_{\mathbf{COL}}}(A)\rangle / \approx_{\Sigma_{\mathbf{COL}},A}$, and it is written also as $BB_{\Sigma_{\mathbf{COL}}}(A)$.*

The main properties of the black box algebra are summarized below, and their proof can be taken from [BH06a].

Proposition 2.21 (Black box properties)**.** *Let $\Sigma_{\mathbf{COL}}$ be a* **COL***-signature and A be a $\Sigma_{\mathbf{COL}}$-algebra. Then $BB_{\Sigma_{\mathbf{COL}}}(A)$ is a reachable and fully-abstract algebra which is* **COL***-isomorphic to A. Moreover, $BB_{\Sigma_{\mathbf{COL}}}(A)$ satisfies literally all the sentences that are satisfied behaviorally by A, i.e. $BB_{\Sigma_{\mathbf{COL}}}(A) \models e$ if and only if $A \models_{\Sigma_{\mathbf{COL}}} e$.*

So, behavioral satisfaction can be reduced to literal satisfaction for the black box algebras. Hence, for the case of two isomorphic **COL** algebras, which have basically the same black box algebra, the sets of behaviorally satisfied sentences are the same.

Corollary 2.22 (**COL**-isomorphic algebras satisfy the same sentences)**.** *Let $\Sigma_{\mathbf{COL}}$ be a* **COL***-signature and A, B be two $\Sigma_{\mathbf{COL}}$-algebras such that $A \equiv_{\Sigma_{\mathbf{COL}}} B$. Then for all sentences e $A \models_{\Sigma_{\mathbf{COL}}} e$ if and only if $B \models_{\Sigma_{\mathbf{COL}}} e$.*

In order to present **COL** as an institution we need to define model reducts along signature morphisms. As has been observed many times in the literature (for example in [MG94, BH06b]), reducts along standard signature morphisms of **COL**-algebras are not always well defined as **COL**-algebras, i.e. they do not necessarily preserve the behavioral satisfaction relation. The solution to this problem is to use only signature morphisms that are compatible with the reachability and observability constraints imposed by the signatures.

Definition 2.23 (Horizontal signature morphism)**.** *Let $\Sigma_{\mathbf{COL}}$ and $\Sigma'_{\mathbf{COL}}$ be two* **COL** *signatures. A* **COL**-signature morphism $\sigma_{\mathbf{COL}} : \Sigma_{\mathbf{COL}} \to \Sigma'_{\mathbf{COL}}$ *is a standard signature morphism $\sigma : \Sigma \to \Sigma'$ such that:*

- *if* $op \in OP_{Cons}$ *then* $\sigma(op) \in OP'_{Cons}$
- *if* $op' : w' \to s' \in OP'_{Cons}$ *with* $s' = \sigma(s)$ *then there exists* $op : w \to s \in OP_{Cons}$ *such that* $\sigma(op) = op'$
- *if* $(op, i) \in OP_{Obs}$ *then* $(\sigma(op), i) \in OP'_{Obs}$
- *if* $(op', i) : w' \to s' \in OP'_{Obs}$ *with* $s'_i = \sigma(s_i)$ *then there exists* $(op, i) : w \to s \in OP_{Obs}$ *such that* $\sigma(op) = op'$ *(s_i is found in w at position i).*

It was proved in [BH06a] that the satisfaction condition holds w.r.t. behavioral satisfaction for horizontal signature morphisms. Hence we can present **COL** as an institution with the basic constituents being **COL**-signatures and horizontal signature morphisms (called **COL**-signature morphisms), **COL**-algebras and first order formulas.

We will now turn our attention to a class of signature morphisms that are not so well behaved, but still rich in properties: the vertical signature morphisms. Vertical signature morphisms can add new constructors and observers and their reducts are useful for information hiding, i.e. translating models that are richer and can be observed in more detail into models that are more restrictive and more abstract. They were discussed in connection with refinement in papers like [MG94] by Malcolm and Goguen and reinterpreted in the setting of **COL** by Bidoit and Hennicker in [BH06b].

Definition 2.24 (Vertical signature morphism)**.** *Let* $\Sigma_{\mathbf{COL}}$ *and* $\Sigma'_{\mathbf{COL}}$ *be two* **COL***-signatures. A* vertical signature morphism σ *between* $\Sigma_{\mathbf{COL}}$ *and* $\Sigma'_{\mathbf{COL}}$ *is a signature morphism* $\sigma : \Sigma \to \Sigma'$ *such that* $\sigma(S_{Obs}) \subseteq S'_{Obs}$ *and* $\sigma(S_{Loose}) \subseteq S'_{Loose}$.

We will now prove some small results concerning the properties of vertical signature morphisms. Some of them are known already for **COL** and some might be known for frameworks akin to **COL**.

Proposition 2.25 (Smaller generated algebras for vertical reducts)**.** *Let* $\Sigma_{\mathbf{COL}}$ *and* $\Sigma'_{\mathbf{COL}}$ *be two* **COL***-signatures and* σ *be a signature morphism between them such that* $\sigma(S_{Loose}) \subseteq S'_{Loose}$. *Consider a* Σ'*-algebra* A' *and let* $A = A'|_\sigma$. *Then* $\langle Gen_{\Sigma_{\mathbf{COL}}}(A)\rangle \subseteq \langle Gen_{\Sigma'_{\mathbf{COL}}}(A')\rangle|_\sigma$.

Proof. If $s \in S_{Loose}$ then $\sigma(s) \in S'_{Loose}$ and $Gen_{\Sigma_{\mathbf{COL}}}(A)_s = A_s = A'_{\sigma(s)} = Gen_{\Sigma'_{\mathbf{COL}}}(A')_{\sigma(s)} = (Gen_{\Sigma'_{\mathbf{COL}}}(A')|_\sigma)_s$.

If $s \in S_{Cons}$ let $X = (X_s)_{s \in S_{Loose}}$ be an infinite set of variables. Then for any term $t \in T_\Sigma(X)_s$ we have a corresponding Σ'-term $\sigma(t)$. This ensures that $\langle Gen_{\Sigma_{\mathbf{COL}}}(A)\rangle \subseteq \langle Gen_{\Sigma_{\mathbf{COL}}}(A')\rangle\restriction_\sigma$. □

Proposition 2.26 (Coarser observational equality for vertical reducts)**.** *Let* $\Sigma_{\mathbf{COL}}$ *and* $\Sigma'_{\mathbf{COL}}$ *be two* **COL***-signatures and* σ *be a vertical signature morphism between them. Consider a* $\Sigma'_{\mathbf{COL}}$*-algebra* A' *and let* A *be a* Σ*-algebra such that* $A = A'\restriction_\sigma$*. Then* $\approx_{\Sigma'_{\mathbf{COL}},A'}\restriction_\sigma \subseteq \approx_{\Sigma_{\mathbf{COL}},A}$.

Proof. Let $\approx = \approx_{\Sigma_{\mathbf{COL}},A}$ and $\approx' = \approx_{\Sigma'_{\mathbf{COL}},A'}\restriction_\sigma$. We want to show that $\approx' \subseteq \approx$. For this consider $s \in S_{State}$ such that $\sigma(s) \in S'_{State}$, $a,b \in A'_{\sigma(s)} = A_s$ such that $a \approx'_s b$ and any context $ctx \in \mathcal{C}(\Sigma_{\mathbf{COL}})_{s\to v}$. We will show that for every valuation α we have that $A_{ctx}[\alpha](a) = A_{ctx}[\alpha](b)$. For such a context $ctx \in \mathcal{C}(\Sigma_{\mathbf{COL}})_{s\to v}$ consider its translation $\sigma(ctx)$ via the signature morphism σ. Because $a \approx' b$ we have that $A'_{\sigma(ctx)}[\alpha](a) = A'_{\sigma(ctx)}[\alpha](b)$. However, from the definition of model reducts we can easily see that $A_{ctx}[\alpha](x) = A'_{\sigma(ctx)}[\alpha](x)$ for all appropriate values x and valuations α. Finally, we can conclude that $a \approx b$ as they yield equal results under all observations. □

Proposition 2.27 (One way translation of **COL**-satisfaction)**.** *Let* $\Sigma_{\mathbf{COL}}$ *and* $\Sigma'_{\mathbf{COL}}$ *be two* **COL***-signatures and* σ *be a vertical signature morphism between them. Then for all* $\Sigma'_{\mathbf{COL}}$*-algebras* A' *and all sentences* $e = \forall X.l = r$ *we have that* $A' \models_{\Sigma_{\mathbf{COL}}} \sigma(e)$ *implies* $A'\restriction_\sigma \models_{\Sigma_{\mathbf{COL}}} e$.

Proof. This is an immediate consequence of Proposition 2.25 and Proposition 2.26 as the set over which quantification is evaluated in smaller in $\Sigma_{\mathbf{COL}}$ and the observational equality is coarser. □

The most important property of vertical signature morphisms w.r.t. their usage for refinements is the fact that they preserve **COL**-isomorphisms. See the proof of the following proposition in [BH05].

Proposition 2.28 (Preservation of **COL**-isomorphisms)**.** *Let* $\Sigma_{\mathbf{COL}}$ *and* $\Sigma'_{\mathbf{COL}}$ *be two* **COL***-signatures and* σ *be a vertical signature morphism between them. For any two* $\Sigma'_{\mathbf{COL}}$*-algebras* A', B' *such that* $A' \equiv_{\Sigma'_{\mathbf{COL}}} B'$ *if* $A'\restriction_\sigma$ *is a* $\Sigma_{\mathbf{COL}}$*-algebra then* $B'\restriction_\sigma$ *is a* $\Sigma_{\mathbf{COL}}$*-algebra and* $A'\restriction_\sigma \equiv_{\Sigma_{\mathbf{COL}}} B'\restriction_\sigma$.

The previous property makes vertical signature morphisms appropriate for describing refinements in the sense of the definitions presented below (see Definition 2.31).

2.3.1 Specifications

Typically a specification has a syntactic part represented by its signature and a semantic part represented by its class of models. The formal theory of specifications and of the operators used to define them can be made at the general level of institutions as in [ST88a].

Definition 2.29 (Basic specifications)**.** *A* **COL**-specification $SP_{\mathbf{COL}}$ *describes a* **COL**-*signature* $\Sigma_{\mathbf{COL}}$ *denoted by* $\mathsf{Sig}[SP_{\mathbf{COL}}]$ *and a class of* $\Sigma_{\mathbf{COL}}$-*algebras denoted by* $\mathsf{Mod}[SP_{\mathbf{COL}}]$. *Basic specifications* $SP_{\mathbf{COL}} = (\Sigma_{\mathbf{COL}}, Ax)$ *consist of a signature* $\Sigma_{\mathbf{COL}}$ *and a set of* Σ-*sentences* Ax *that give the semantics* $\mathsf{Mod}[SP_{\mathbf{COL}}] = \mathsf{Alg}_{\mathbf{COL}}(\Sigma_{\mathbf{COL}}, Ax)$.

Similar definitions can be given for standard specifications, written SP, by using standard algebraic signatures for representing the syntax and standard algebras with standard satisfaction for the semantics.

Please note that by abuse of notation we will sometimes write $A \in SP_{\mathbf{COL}}$ to denote a model in the semantics of $SP_{\mathbf{COL}}$, i.e. $A \in \mathsf{Mod}[SP_{\mathbf{COL}}]$. Also, many times we assume implicitly that the signature of $SP_{\mathbf{COL}}$ is $\Sigma_{\mathbf{COL}}$, or that the signature of $SPI_{\mathbf{COL}}$ is $\Sigma I_{\mathbf{COL}}$, etc.

During this thesis we will use operators for building structured specifications [BG80, SB83].

Definition 2.30 (Structured specifications)**.** *Let* $SP_{\mathbf{COL}}$ *be a* **COL**-*specification,* $\Sigma'_{\mathbf{COL}}$ *be a* **COL**-*signature and* $\phi : \Sigma \to \Sigma'$ *be a standard signature morphism. We denote by* $\phi(SP_{\mathbf{COL}})$ *the* translated specification along ϕ *for which we have* $\mathsf{Sig}[\phi(SP_{\mathbf{COL}})] = \Sigma'_{\mathbf{COL}}$ *and* $\mathsf{Mod}[\phi(SP_{\mathbf{COL}})] = \{A' \in \mathsf{Mod}(\Sigma_{\mathbf{COL}}) \mid A'|_\phi \in \mathsf{Mod}[SP_{\mathbf{COL}}]\}$. *Let* $SP^0_{\mathbf{COL}}$ *and* $SP^1_{\mathbf{COL}}$ *be two* **COL** *specifications on the same* **COL**-*signature. We denote by* $SP^0_{\mathbf{COL}} + SP^1_{\mathbf{COL}}$ *the* sum specification *for which we have* $\mathsf{Sig}[SP^0_{\mathbf{COL}} + SP^1_{\mathbf{COL}}] = \mathsf{Sig}[SP^0_{\mathbf{COL}}] = \mathsf{Sig}[SP^1_{\mathbf{COL}}]$ *and* $\mathsf{Mod}[SP^0_{\mathbf{COL}} + SP^1_{\mathbf{COL}}] = \mathsf{Mod}[SP^0_{\mathbf{COL}}] \cap \mathsf{Mod}[SP^1_{\mathbf{COL}}]$.

In the following lines we will give examples that will show the way we will present specifications.

spec BOOL
  sorts *Bool*
  operations *true*, *false* : *Bool*
  constructors *true*, *false*
  axioms $true \neq false$
end

spec NAT
  sorts Nat
  operations $0 : Nat$
    $s : Nat \to Nat$
  constructors $0, s$
  axioms $\forall x, y : Nat$
    $s(x) = s(y) \longrightarrow x = y$
    $s(x) \neq 0$
end

Above we have described two specifications, BOOL and NAT, each of them having a constrained visible sort. We will shorten the presentation of axioms by writing the variables just once and considering that each axiom is universally quantified. However, when we need to write more complex axioms we will be more explicit about the quantification and the scope of variables.

Typically the nature of sorts will be directly inferred from the distribution of constructors and observers. This is done in conformance with the style of presentation imposed in [BH06a].

spec IMPLICITSAMPLE **THEN**
  sorts s, v
  operations $a : s$
    $b : v$
    $obs : s \to v$
  constructors a, b
  observers obs
end

We can analyze IMPLICITSAMPLE and infer that s is a hidden sort because obs is an observer for it and both s, v are constructed because they have at least one constructor. However, in some cases we might need to take advantage of the flexibility given by the framework established in Remark 2.11, and in order to specify explicitly that a sort is constructed or hidden we use the following syntax:

```
spec EXPLICITSAMPLE = BOOL + NAT THEN
  sorts s, v
  hidden s
  constructed v
end
```

The specification EXPLICITSAMPLE has a hidden loose sort s and a constrained visible sort v. Accordingly its algebras will have an empty generated part on the visible sort and a total observational equality on the hidden sort.

Notice that we have also used operators to build structured specifications. The keyword **THEN** denotes a translation specification followed by some additional requirements written in the body of the currently defined specification.

To specify higher order types we will typically use *combinatorial specifications*, i.e. specifications that have a combinatorial signature and also satisfy the corresponding combinatorial axioms. The requirement that all combinators to be defined and all corresponding sorts to be included in the signature will appear simply written as the word **combinatorial** in the specification.

2.3.2 Refinement of behavioral specifications

A behavioral refinement $SP_{\mathbf{COL}} \rightsquigarrow SPI_{\mathbf{COL}}$ represents the process of implementing a target specification $SP_{\mathbf{COL}}$ by a more concrete source specification $SPI_{\mathbf{COL}}$ via a construction that will map models of $SPI_{\mathbf{COL}}$ to models of $SP_{\mathbf{COL}}$. Behavioral refinements have been studied extensively in the literature [Sch87, ST88b] and the discussion was actualized to **COL** in [BH05, BH06b].

As we have said the notion of refinement is based on that of construction which is defined as follows:

Definition 2.31 (Construction)**.** *Let $SP_{\mathbf{COL}}$ and $SPI_{\mathbf{COL}}$ be two* **COL***-specifications. A* **COL** construction *k between $SPI_{\mathbf{COL}}$ and $SP_{\mathbf{COL}}$, written $k : SPI_{\mathbf{COL}} \Rightarrow SP_{\mathbf{COL}}$, is a function $k : \mathsf{Mod}[SPI_{\mathbf{COL}}] \to \mathsf{Mod}[SP_{\mathbf{COL}}]$ that preserves* **COL***-isomorphisms, i.e. for all $AI, BI \in \mathsf{Mod}[SPI_{\mathbf{COL}}]$ such that $AI \equiv_{\Sigma I_{\mathbf{COL}}} BI$ we have that $k(AI) \equiv_{\Sigma_{\mathbf{COL}}} k(BI)$. We call the pair of specifications, $SPI_{\mathbf{COL}}$ and $SP_{\mathbf{COL}}$, the* context *of the construction.*

The definition we use here is slightly different than the one used in [BH05, BH06b]. The main difference is that we treat directly constructions between specifications rather

than just between signatures, and hence we will work only with constructions that are total functions.

A refinement is a relation between specifications that amounts to the existence of a construction. Basically, we say that $SPI_{\mathbf{COL}}$ is a refinement of $SP_{\mathbf{COL}}$ along k if there exists a construction $k : SPI_{\mathbf{COL}} \Rightarrow SP_{\mathbf{COL}}$.

As it appears from Proposition 2.28, reducts along vertical signature morphisms provide **COL**-constructions and are suitable for refinements when they are well defined.

A special kind of constructions are those for which the signature of the source specification is "embedded" in that of the target specification, and the construction preserves the interpretation of the symbols from the source signature. These constructions are called *persistent* constructions.

Definition 2.32 (Persistent construction)**.** *Let* $\Sigma I_{\mathbf{COL}}$ *and* $\Sigma_{\mathbf{COL}}$ *be two* **COL***-signatures and* $\sigma_{\mathbf{COL}} : \Sigma I_{\mathbf{COL}} \to \Sigma_{\mathbf{COL}}$ *be a signature morphism between them. A construction* $k : SPI_{\mathbf{COL}} \Rightarrow SP_{\mathbf{COL}}$ *is* persistent *if for all* $AI \in SPI_{\mathbf{COL}}$ *we have that* $k(AI)\restriction_{\sigma_{\mathbf{COL}}} = AI$.

In order to simplify the presentation we will sometimes omit to explicitly name the underlying signature morphism for a persistent construction, but implicitly assume the default name $\sigma_{\mathbf{COL}}$ for a construction k, or $\sigma'_{\mathbf{COL}}$ for k'.

The next definition characterizes the class of constructions that will be used throughout this thesis. We will investigate properties of constructions that implement new operations defined on old sorts, i.e. which are persistent along a tight signature morphism (see Section 2.2).

Definition 2.33 (Tight construction)**.** *Let* $k : SPI_{\mathbf{COL}} \Rightarrow SP_{\mathbf{COL}}$ *be a persistent construction. We call* k *a* tight construction *if the underlying signature morphism* $\sigma_{\mathbf{COL}} : \Sigma I_{\mathbf{COL}} \to \Sigma_{\mathbf{COL}}$ *is a tight signature morphism.*

One of our objectives for the chapters that follow is to prove definability properties for some special kinds of constructions. i.e. that operations added in Σ' are definable in terms of those present in Σ. However, this problem makes most sense when the sets of sorts are the same in both signatures, and this is the reason why the tight constructions will stand at the basis of our investigation.

2.4 Other behavioral frameworks

Some of the problems that will be investigated in this thesis in the framework of **COL** have been investigated in a simpler setting, that will be denoted by **OBS**. Basically the study of stability of constructions and the introduction of the technique of lifting constructions were developed in [Sch87, San99, HB99, BST08] relative to a framework which can be seen as a "sub-institution" of **COL** containing only a particular subclass of **COL** signatures. These signature will be called *integral signatures*, and they have the distinguishing feature that all operations that produce or consume state sorts are necessarily constructors and respectively observers.

Definition 2.34 (Integral **COL**-signature)**.** *Let* $\Sigma_{\mathbf{COL}}$ *be a* **COL** *signature. We say that* $\Sigma_{\mathbf{COL}}$ *is integral if*

- $S_{Loose} = S_{Obs}$ *and* $S_{Cons} = S_{State}$
- *for every* $op : s_1, \ldots, s_n \to s$ *such that* $s \in S_{State}$ *we have that* $op \in OP_{Cons}$
- *for every* $op : s_1, \ldots, s_n \to s$ *and* $i \in 1 \ldots n$ *such that* $s_i \in S_{State}$ *we have that* $(op, i) \in OP_{Obs}$.

Please notice that in an integral signature there are only two kinds of sorts: visible (which are also loose) and hidden (which are also constructed).

The following signature is an example of an integral signature:

spec INTEGRALSAMPLE **THEN**
  sorts s, v
  operations $a : s$
    $op : s \to s$
    $obs : s, s \to v$
    $obs' : s \to s$
  constructors a, op
  observers $(obs, 1), (obs, 2), obs'$
end

In the signature of INTEGRALSAMPLE loose sorts are visible and constructed sorts are hidden. More importantly all the operations that have a hidden result sort, i.e. a and op, are constructors. And, all operations that have a hidden sort in the argument list are observers (for all positions in the arguments list of the hidden sort).

Please notice that if there is at least one plain operation that produces or consumes a hidden sort that is not among the set of constructors or observers then the signature is not integral. The absence of such plain operations has an important consequence, namely that the standard algebras of integral signatures are necessarily **COL**-algebras, i.e. the reachability constraint and observability constraint are trivially satisfied for integral signatures. That happens firstly because hidden elements produced by operations are, by definition, produced by constructors and therefore in the generated part. Secondly, because observers always commute with observational equality the observability constraints hold for all operations that act on hidden sorts.

Before going any further please notice that integral signatures can be presented without naming explicitly the constructors and the observers. For an operation the quality of being a constructor or an observer can be inferred from the nature of its arguments or result sort. Hence, it is sufficient to describe the set of hidden sorts and of visible sorts in order to get a complete definition of an integral signature. That is the way that was typically chosen in previous presentations like [Mit91, San99, GM00] for describing behavioral aspects of a specification.

We will use the name **OBS** to denote the "sub-institution" of **COL** obtained by only considering the integral signatures. An important observation is that in **OBS** the notion of behavioral isomorphisms between algebras (inherited automatically from **COL**) becomes equivalent with the notion of observational equivalence used in [Sch87, Mit91, San99, BST08]. More explicitly, one can see that two algebras of an integral signature are **COL**-isomorphic if and only if all ground terms of a visible sort evaluate to the same thing. We will reproduce the definition of observational equivalence from [ST87] to make the concept clear.

Definition 2.35 (Observational equivalence)**.** *Let $\Sigma_{\mathbf{COL}}$ be an integral* **COL***-signature. Let A and B be two $\Sigma_{\mathbf{COL}}$-algebras that are equal on the visible sorts. We say that A and B are* observational equivalent*, written $A \equiv_{Obs} B$, if for any set X with variables of visible sorts and any term $t \in T_\Sigma(X)$ of visible sort, for each valuation $\alpha : X \to A$ (so that $\alpha : X \to B$ as well) we have that $A_t[\alpha] = B_t[\alpha]$*

It is easy to prove that two algebras are observationally equivalent if and only if there exists a **COL**-isomorphism that is identity on the visible sorts. The remark about the coincidence between the two concepts, observational equivalence and **COL**-isomorphism, was first made in [BH06a].

Another remark is that in **OBS** the conditions under which a signature morphism

is vertical reduce to a single requirement, i.e. that visible sorts are preserved (which of course means that the loose sorts are preserved as well). This observation is especially interesting when seen in conjunction with the study presented in the following chapters. Our investigation of lifted constructions in **COL** is similar to the one done in [Sch87, BST08] for **OBS**. But, because the morphisms used for **OBS** were those that preserve visible sorts, which in **COL**-terminology are the vertical ones, it means that previous presentations of the subject are a particular case of the presentation we give in Chapter 4.

2.5 Externalized behavioral semantics

We have seen until now, the presentation of what is called the internalized behavioral semantics. In this section we plan to look at the externalized way of defining behavioral semantics. In [BH05] Bidoit and Hennicker made a comprehensive comparison of the two variants of presenting behavioral semantics.

The internalized behavioral semantics is usually defined by setting a logical framework that has a built in notion of behavioral satisfaction. With the help of the behavioral satisfaction relation one can relax the requirements for models in the semantics of a specification, by considering as acceptable implementations not only the algebras that satisfy literally some axioms but all those that satisfy them behaviorally. The presentation of **COL** illustrates this pattern, and hence the semantics of **COL** basic specifications is given with respect to **COL**-satisfaction.

However, in the literature, one can find different ways to enhance the notion of an acceptable model for a specification. The main alternative to the internalized view for defining behavioral semantics is the externalized view, which is based on abstraction operators. Abstraction operators transform classes of algebras representing the standard semantics into classes of algebras which represent the behavioral semantics. We will see below the definition for some of these operators.

Definition 2.36 (Abstraction operators)**.** *Let* $\Sigma_{\mathbf{COL}}$ *be a* **COL***-signature and* $\mathcal{A}$ *be a class of* Σ*-algebras. The* behavioral extension operator *is defined as follows:*

$$Beh_{\Sigma_{\mathbf{COL}}}(\mathcal{A}) = \{A \in \mathsf{Alg}_{\mathbf{COL}}(\Sigma_{\mathbf{COL}}) \mid BB_{\Sigma_{\mathbf{COL}}}(A) \in \mathcal{A}\}$$

The abstractor operator *is defined as follows:*

$$Iso_{\Sigma_{\mathbf{COL}}}(\mathcal{A}) = \{A \in \mathsf{Alg}_{\mathbf{COL}}(\Sigma_{\mathbf{COL}}) \mid \textit{there exists } B \in \mathcal{A} \textit{ such that } A \equiv_{\Sigma_{\mathbf{COL}}} B\}$$

The reachable and fully-abstract operator *is defined as follows:*

$$RFA_{\Sigma_{\mathbf{COL}}}(\mathcal{A}) = \{A \in \mathsf{Alg}_{\mathbf{COL}}(\Sigma_{\mathbf{COL}}) \mid A \in \mathcal{A} \textit{ is reachable and fully-abstract w.r.t. } \Sigma_{\mathbf{COL}}\}$$

Definition 2.37 (Behaviorally closed classes)**.** *A class of* Σ*-algebras* $\mathcal{A}$ *is called* behaviorally closed *w.r.t.* $\Sigma_{\mathbf{COL}}$ *if* $\mathcal{A} \subseteq Beh_{\Sigma_{\mathbf{COL}}}(\mathcal{A})$.

Before starting to present how these operators are used to define the externalized behavioral semantics of a specification we shall enumerate some useful properties of the abstraction operators.

Proposition 2.38 (Abstraction operators properties)**.** *Let* $\Sigma_{\mathbf{COL}}$ *be a* **COL***-signature and* $\mathcal{A}$ *a class of* Σ*-algebras that is closed under standard isomorphisms. Then:*

- $Beh_{\Sigma_{\mathbf{COL}}}$ *,* $Iso_{\Sigma_{\mathbf{COL}}}$ *and* $RFA_{\Sigma_{\mathbf{COL}}}$ *are idempotent and monotonic*
- $RFA_{\Sigma_{\mathbf{COL}}}(\mathcal{A}) \subseteq \mathcal{A} \subseteq Iso_{\Sigma_{\mathbf{COL}}}(\mathcal{A})$
- $Beh_{\Sigma_{\mathbf{COL}}}(\mathcal{A}) \subseteq Iso_{\Sigma_{\mathbf{COL}}}(\mathcal{A})$
- $Beh_{\Sigma_{\mathbf{COL}}}(\mathcal{A}) = Iso_{\Sigma_{\mathbf{COL}}}(RFA_{\Sigma_{\mathbf{COL}}}(\mathcal{A}))$
- $Beh_{\Sigma_{\mathbf{COL}}}(\mathcal{A}) = Iso_{\Sigma_{\mathbf{COL}}}(Beh_{\Sigma_{\mathbf{COL}}}(\mathcal{A}))$
- $Beh_{\Sigma_{\mathbf{COL}}}(\mathcal{A}) = Iso_{\Sigma_{\mathbf{COL}}}(\mathcal{A})$ *if* $\mathcal{A}$ *is behaviorally closed*

Proof. The most important identity on which the relation between the three operators is based is $Beh_{\Sigma_{\mathbf{COL}}}(\mathcal{A}) = Iso_{\Sigma_{\mathbf{COL}}}(RFA_{\Sigma_{\mathbf{COL}}}(\mathcal{A}))$, so we will only prove this one as the other ones follow immediately.

Let $A \in Beh_{\Sigma_{\mathbf{COL}}}(\mathcal{A})$. This means that $BB_{\Sigma_{\mathbf{COL}}}(A) \in \mathcal{A}$. But from Proposition 2.21 we know that $A \equiv_{\Sigma_{\mathbf{COL}}} BB_{\Sigma_{\mathbf{COL}}}(A)$ and also that $BB_{\Sigma_{\mathbf{COL}}}(A)$ is reachable and fully-abstract and hence we get $A \in Iso_{\Sigma_{\mathbf{COL}}}(RFA_{\Sigma_{\mathbf{COL}}}(\mathcal{A}))$.

Let $A \in Iso_{\Sigma_{\mathbf{COL}}}(RFA_{\Sigma_{\mathbf{COL}}}(\mathcal{A}))$. We get that there exists $B \in \mathcal{A}$ that is reachable and fully-abstract such that $A \equiv_{\Sigma_{\mathbf{COL}}} B$, and furthermore that $BB_{\Sigma_{\mathbf{COL}}}(A) \equiv_{\Sigma_{\mathbf{COL}}} B$. Because $BB_{\Sigma_{\mathbf{COL}}}(A)$ and B are both reachable and fully-abstract we get that they are standardly isomorphic and hence $BB_{\Sigma_{\mathbf{COL}}}(A) \in \mathcal{A}$. □

2.5.1 Appropriate semantics for basic specifications

The externalized view of the semantics of behavioral specifications was considered in the literature in two main forms. The first flavor, which is encountered in [ST87], is

based on the abstractor operator and treats as acceptable denotations of a specification all models that are observational equivalent with a standard model of the specification. The second flavor, encountered in [BH98], is based on the behavioral extension operator and accepts all models whose black box behavior is a standard model of the specification. We will use the adapted versions of these operators to **COL** as it is done in [BHW95, BH05]. The $Iso_{\Sigma_{\mathbf{COL}}}$ operator is the abstractor operator in **COL** and applied to a class of Σ-algebras it produces the class of **COL**-algebras that are observationally equivalent to those in the original class (recall that observational equivalence from **OBS** is the same as isomorphism in **COL**). The $Beh_{\Sigma_{\mathbf{COL}}}$ operator is the behavioral extension operator and applied to a class of Σ-algebras it produces the class of all the **COL**-algebras that have their black box behavior in the original class.

These operators are used to give behavioral semantics starting from classes of standard algebras. In particular they can be used to give the behavioral semantics of basic specifications by applying the operators to the class of algebras that literally satisfy the sentences.

Definition 2.39 (Externalized behavioral semantics)**.** *Let* $\Sigma_{\mathbf{COL}}$ *be a* **COL***-signature and* $Ax \subseteq \mathsf{Sen}(\Sigma)$ *be a set of* Σ*-sentences.*

The abstractor semantics *for the basic specification* $SP_{\mathbf{COL}} = (\Sigma_{\mathbf{COL}}, Ax)$ *is given as follows:* $\mathsf{Mod}_a[SP_{\mathbf{COL}}] = Iso_{\Sigma_{\mathbf{COL}}}(\mathsf{Alg}(\Sigma, Ax))$.

The behavior extension semantics *for the basic specification* $SP_{\mathbf{COL}} = (\Sigma_{\mathbf{COL}}, Ax)$ *is given as follows:* $\mathsf{Mod}_b[SP_{\mathbf{COL}}] = Beh_{\Sigma_{\mathbf{COL}}}(\mathsf{Alg}(\Sigma, Ax))$.

It is shown in [BHW95] that the abstractor operator and the behavioral extension operator coincide on behaviorally closed classes (the result is for **OBS** but it can be easily adapted to **COL** using Proposition 2.38). Classes that are not behaviorally closed were deemed to be uninteresting or inconsistent and were neglected (quote from [BHW95]):

> A specification *SP* is behaviorally consistent if the behavior of any model of *SP* is also a model of *SP* (and hence fulfills the requirements of *SP*)

The original name for behavioral closedness was *behavioral consistency*. But that changed quickly in [BH96], where we encounter the following paragraph:

> An algebraic specification *SP* is behaviorally closed if the behaviors of all models of *SP* are also models of *SP*. If this is not the case, this means that there is some "inconsistency" between the properties required by the specification *SP* and the chosen behavioral equality.

and in a footnote a reference is made to the previous definition in [BHW95]:

> "Behaviorally closed" was called "behaviorally consistent" in [BHW95], but the terminology proposed here seems more adequate.

Despite that partial recognition of the fact that behavioral closedness should not be taken for behavioral consistency, the bias against classes that are not behaviorally closed has been propagated since. Therefore, because almost always the presentations were assuming behaviorally closed classes the distinction between the two kinds of semantics is blurred in the literature. In this section we propose an adequate definition for consistency of behavioral specifications, and by rehabilitating the classes that are not behaviorally closed we will make some comments on the difference between the two ways of defining the externalized behavioral semantics.

First let us look at what behavioral closedness requires in our present framework and whether this notion can be a sensible definition of behavioral consistency. Definition 2.37 says that a class of Σ-algebras $\mathcal{A}$ is behaviorally closed if $\mathcal{A} \subseteq Beh_{\Sigma_{\mathbf{COL}}}(\mathcal{A})$. That means in particular that all algebras in $\mathcal{A}$ are **COL**-algebras. This is a very strong requirement when analyzed with respect to a basic specification $(\Sigma_{\mathbf{COL}}, Ax)$. It implies that axioms Ax are enough to enforce the **COL** constraints when interpreted literally, i.e. taking $\mathcal{A} = \mathsf{Alg}(\Sigma, Ax)$. It is obvious that such a requirement cannot be the test of consistency in **COL** as virtually all basic specifications will have to be regarded as behaviorally inconsistent. This obvious problem could not have been noticed in the original framework of **OBS** used in [BHW95] and [BH96], where the corresponding **COL**-constraints were trivially satisfied (see the discussion on integral signatures in Section 2.4). But even if we assume that our basic specification is powerful enough to enforce the **COL** constraints, by using integral signatures as in **OBS** or by adding additional axioms, we still don't agree that behavioral closedness is appropriate as a notion of consistency. For that, let us consider the original example (Example 3.18 in [BHW95]) that was used to illustrate a behaviorally inconsistent specification.

spec DEMO
  sorts s
  operations $a, b : s$
  hidden s
  axioms $a \neq b$
end

We agree that the specification DEMO is inconsistent and that the class of models that literally satisfy $a \neq b$ is not behaviorally closed. The arguments are the same as in [BHW95], namely that because there are no observers, the observational equality on the sort s relates a, b, and therefore there is no literal model of $a \neq b$ that behaviorally satisfies $a \neq b$. What we do not agree with is that inconsistency is caused by the fact that the class of models that literally satisfy $a \neq b$ is not behaviorally closed. We can look at another example for which the same cause could be invoked but where it seems inappropriate to do so, and we should consider the specification as being consistent.

spec DEMO+
  sorts s, v
  operations $a, b : s$
    $obs : s \to v$
  observers obs
  axioms $a \neq b$
end

Now, the class of literal models of specification DEMO+ is still not behaviorally closed. To see that, recall that a class $\mathcal{A}$ is behaviorally closed if for all $A \in A$ the black box $BB_{\Sigma_{\mathbf{COL}}}(A) \in \mathcal{A}$. Therefore, in order to show that the literal models of $a \neq b$ do not form a behaviorally closed class, we must find a literal model of $a \neq b$ such that its black box does not satisfy $a \neq b$. We can easily imagine such a model that interprets differently a and b literally satisfying $a \neq b$, but for which the observation obs produces equal results on a and b, forcing its black box to satisfy $a = b$. However, there are many models for which the observation produces different results and for which the black box satisfies $a \neq b$, and there is no reason why we should not accept those as part of the behavioral semantics of DEMO+.

To sum up, we have seen two examples DEMO and DEMO+ for which the literal class of models is not behaviorally closed but we think that one should be considered behaviorally inconsistent and the other one behaviorally consistent. In order to formalize these insights we will give the definition of consistency in the externalized style of behavioral semantics. Basically we say that a class of standard algebras is behaviorally consistent if it has at least one reachable and fully abstract algebra.

Definition 2.40 (Behavioral consistency)**.** *Let* $\Sigma_{\mathbf{COL}}$ *be a* **COL***-signature. A class* $\mathcal{A}$

of Σ*-algebras is* behaviorally consistent w.r.t. $\Sigma_{\mathbf{COL}}$ *if* $RFA_{\Sigma_{\mathbf{COL}}}(\mathcal{A}) \neq \emptyset$. *A set* E *of* Σ*-sentences is called* behaviorally consistent w.r.t. $\Sigma_{\mathbf{COL}}$ *if* $\mathsf{Alg}(\Sigma, E)$ *is behaviorally consistent w.r.t.* $\Sigma_{\mathbf{COL}}$.

We can now explain why DEMO is inconsistent: because there is no fully abstract model that literally satisfies the required axiom $a \neq b$; all fully abstract models will satisfy literally $a = b$. On the other hand there are fully abstract models of $a \neq b$ w.r.t. the signature of DEMO+ and hence that specification is behaviorally consistent.

Now that we have explained why one should accept behaviorally non-closed classes as consistent specifications, we should discriminate between the two choices of externalized behavioral semantics: the abstractor semantics vs. the behavioral extension semantics. We announce our preference for the behavioral extension semantics and we justify that by its compatibility with the internalized behavioral semantics. In other words the choice is made such that the externalized behavioral semantics and the internalized behavioral semantics of a basic specification coincide and that happens because:

$$Beh_{\Sigma_{\mathbf{COL}}}(\mathsf{Alg}(\Sigma, Ax)) = \mathsf{Alg}_{\mathbf{COL}}(\Sigma_{\mathbf{COL}}, Ax)$$

The previous identity can be easily proved by using the fact that each algebra satisfies behaviorally an axiom if and only if its black box satisfies it literally (see Proposition 2.21).

One of the reasons for preferring a behavioral semantics that is in accordance with the **COL** satisfaction relation, is that the deduction system for behavioral specifications will have the good properties of an abstract entailment system in the sense of Meseguer [Mes89]. In contrast, one can observe the lack of one important property for the deduction system $\vdash^{Iso}$ induced by the $Iso_{\Sigma_{\mathbf{COL}}}$ operator, defined bellow.

Definition 2.41 (Behavioral deduction systems)**.** *Let* $\Sigma_{\mathbf{COL}}$ *be a* **COL***-signature.*

We define the deduction system $\vdash^{Beh}$ *as a relation between sets of* $\Sigma_{\mathbf{COL}}$*-sentences as follows:* $Ax_0 \vdash^{Beh} Ax_1$ *if* $Beh_{\Sigma_{\mathbf{COL}}}(\mathsf{Alg}(\Sigma, Ax_0)) \subseteq Beh_{\Sigma_{\mathbf{COL}}}(\mathsf{Alg}(\Sigma, Ax_1))$.

Similarly, we define the deduction system $\vdash^{Iso}$ *as a relation between sets of* $\Sigma_{\mathbf{COL}}$*-sentences as follows:* $Ax_0 \vdash^{Iso} Ax_1$ *if* $Iso_{\Sigma_{\mathbf{COL}}}(\mathsf{Alg}(\Sigma, Ax_0)) \subseteq Iso_{\Sigma_{\mathbf{COL}}}(\mathsf{Alg}(\Sigma, Ax_1))$.

As we have already anticipated, $\vdash^{Beh}$ is well-behaved and it has the property of allowing a proof goal to be split into subgoals.

Proposition 2.42 ($\vdash^{Beh}$ has unions)**.** *Let* Σ *be an algebraic signature and let* Ax, Ax_1 *be two sets of* Σ*-sentences. Then* $Ax \vdash^{Beh} Ax_0 \cup Ax_1$ *if* $Ax \vdash^{Beh} Ax_0$ *and* $Ax \vdash^{Beh} Ax_1$.

Proof. By considering the coincidence with the internalized semantics given by **COL**-satisfaction, the goal is equivalent to: $Ax \models_{\mathbf{COL}} Ax_0 \cup Ax_1$ if $Ax \models_{\mathbf{COL}} Ax_0$ and $Ax \models_{\mathbf{COL}} Ax_1$; and this is trivial in any institution. □

Of course $\vdash^{Iso}$ will also have unions when we consider only behaviorally closed sets of sentences, as it coincides with $\vdash^{Beh}$. However, for axioms that have a class of models that is not behaviorally closed the property fails.

Example 2.43 ($\vdash^{Iso}$ does not have unions)**.** *$Ax \vdash^{Iso} Ax_0$ and $Ax \vdash^{Iso} Ax_1$ does not imply $Ax \vdash^{Iso} Ax_0 \cup Ax_1$. Let's consider a* **COL***-signature $\Sigma_{\mathbf{COL}} = (S = \{s, v\}, OP = \{a, b : s, obs : s \to v\}, OP_{Obs} = \{obs\})$. We have that $a = b \vdash^{Iso} a = b$ and $a = b \vdash^{Iso} a \neq b$ but $a = b$ does not entail the union. The crucial step here is that for a Σ-algebra A such that $A \models a = b$, we can find a model B such that $A \equiv_{\Sigma_{\mathbf{COL}}} B$ but $B \models a \neq b$. Simply interpret a and b differently but let the observation produce the same result as in A.*

This is a big drawback as it means that proof goals cannot be split in general into subgoals that can be solved independently, and justifies our preference for the behavioral extension operator over the abstractor operator. Despite the clear advantage of the behavioral extension operator, the abstractor operator is still typically used to give semantics for behavioral specifications and one reason for this is that it is more convenient to express the basic notions of the theory of refinements. However, this is done at the expense of uniformity, i.e. by neglecting classes that are not behaviorally closed or by accepting drawbacks such as the one exposed above. A consequence of that is that most of the work on refinement is done only for behaviorally closed classes, for which the choice between the two kinds of semantics does not matter. In Section 2.5.2 we will see an example of the confusion that can arise from the non-recognition of the validity of behavioral non-closed classes.

2.5.2 Proving correctness of refinements

We saw that the inaccurate definition of behavioral consistency has blurred the demarcation line between the two options for giving externalized semantics for behavioral specifications, even though the choice should have been clear. We will see in this section that this confusion also generated incomplete methods for proving correctness of behavioral refinements, even though their full power was easily accessible.

Consider the case of a **COL**-construction $k : \mathsf{Mod}[SPI_{\mathbf{COL}}] \Rightarrow \mathsf{Mod}[\Sigma]$ that we want to prove is a correct construction between $SPI_{\mathbf{COL}}$ and $SP_{\mathbf{COL}}$. In order to actually

prove the correctness of k we need to show that for any model $AI \in SPI_{\mathbf{COL}}$ we have that $k(AI) \in SP_{\mathbf{COL}}$. If we assume that $SPI_{\mathbf{COL}}$ and $SP_{\mathbf{COL}}$ are basic specifications $(\Sigma I_{\mathbf{COL}}, AxI)$ respectively $(\Sigma_{\mathbf{COL}}, Ax)$ we must show that for all $AI \models_{\Sigma I_{\mathbf{COL}}} AxI$ we have that $k(AI) \models_{\Sigma_{\mathbf{COL}}} Ax$.

Several methods have been proposed for simplifying proofs of correctness for behavioral constructions. One of the most important techniques is applicable whenever the axioms Ax can be translated into the signature ΣI. This is the case in many examples [Ros03, BH06a, BH06b, HLST00]. Typically, this scenario happens when k is a model reduct $\restriction_{\phi}$ along a signature morphism $\phi : \Sigma \to \Sigma I$. This means that we can use ϕ to translate the axioms Ax into ΣI as $\phi(Ax)$. In this setting, we can do the proof of correctness in the signature $\Sigma I_{\mathbf{COL}}$, by showing roughly that $AxI \models_{\Sigma I_{\mathbf{COL}}} \phi(Ax)$. Since the proof obligation is expressible in a single signature, powerful proof methods designed to work directly with observational equality are available like Rosu's circular coinduction method [RG00]. Of course, the situation is not always that simple; as the **COL** constraints imposed by the two signatures can differ significantly, one should take care when doing the translation in order to ensure soundness. The sound translation of Ax might involve the explicit encoding of the $\Sigma_{\mathbf{COL}}$ constraints into the signature $\Sigma I_{\mathbf{COL}}$ [BH06a, BH95], but to illustrate that is not our purpose. It is important to understand that this kind of simplification, reducing the goal to a single signature, works only when we have a sound translation of Ax into ΣI.

For other cases, when such a syntactic translation is not available, a semantic approach is needed in order to reduce the complexity of the task of proving the correctness of the construction. Such semantic simplifications where first proposed by Sannella and Tarlecki in [ST88b] and recently by Bidoit and Hennicker for **COL** in [BH06b]. Before starting to look at these methods we will present an example for which the syntactic approach does not work.

Example 2.44. *Let* $SPI_{\mathbf{COL}} = (\Sigma I_{\mathbf{COL}}, AxI)$ *and* $SP_{\mathbf{COL}} = (\Sigma_{\mathbf{COL}}, Ax)$ *where* $\Sigma I_{\mathbf{COL}} = (\{s, v\}, \{obs : s \to v, a : s\}, \{obs\})$, $\Sigma_{\mathbf{COL}} = (\{s, v\}, \{obs : s \to v, a, b : s\}, \{obs\})$, $AxI = \{\forall x, y. x \neq a \wedge y \neq a \longrightarrow x = y\}$ *and* $Ax = \{(\exists x. x \neq a) \longrightarrow b \neq a\}$.

Let $k : SPI_{\mathbf{COL}} \Rightarrow \mathsf{Alg}_{\mathbf{COL}}(\Sigma_{\mathbf{COL}})$ *be defined as follows:*

- $k(AI)_b$ *is interpreted as* $k(AI)_a$ *if for all* $c \in AI_s$, $AI_{obs}(c) = AI_{obs}(a)$
- $k(AI)_b$ *is interpreted as an element* $c \in A_s$ *such that* $AI_{obs}(c) \neq AI_{obs}(a)$ *otherwise.*

We have a setting in which s, v are both loose sorts, v is visible, and the elements of s are constrained by $SP_{\mathbf{COL}}$ to be either behaviorally equal to a or otherwise equal between them. The construction k implements b as one of the elements different from a if it exists or as a if none exists.

We can see that k takes isomorphic algebras to isomorphic algebras. However, we cannot soundly translate the axioms Ax into $\Sigma I_{\mathbf{COL}}$ because the construction k is based on elements that are not representable as ΣI-terms. Therefore, our only chance to prove correctness of the construction is a semantic proof, i.e. we need to prove that for all $AI \in SPI_{\mathbf{COL}}$ we have that $k(AI) \models_{\Sigma_{\mathbf{COL}}} Ax$.

After this introduction let us focus on the semantic simplification proposed in [ST88b, BH06b]. Because **COL** constructions preserve isomorphisms one could do the proofs of correctness only for the literal models of the specification and then extend this proof automatically to all models of the source specification.

Bidoit and Hennicker propose in [BH06b] the following rule for any **COL** construction k.

$$\frac{k(\mathsf{Alg}(\Sigma I, AxI)) \models_{\Sigma_{\mathbf{COL}}} Ax}{k(\mathsf{Alg}_{\mathbf{COL}}(\Sigma I_{\mathbf{COL}}, AxI)) \models_{\Sigma_{\mathbf{COL}}} Ax} \tag{2.1}$$

In other words, it is sufficient to prove only that models constructed from the literal algebras of AxI satisfy the axioms Ax behaviorally. This rule is sound because any model AI_0 that satisfies AxI behaviorally is **COL**-isomorphic to a model AI_1 that satisfies AxI literally , namely its black box behavior. Furthermore because k preserves isomorphisms we get that $k(AI_0) \equiv_{\Sigma_{\mathbf{COL}}} k(AI_1)$ and using the assumption that $k(AI_1) \models_{\Sigma_{\mathbf{COL}}} Ax$ we also get that $k(AI_0) \models_{\Sigma_{\mathbf{COL}}} Ax$.

Two points need to be made here. First point is that the rule is sound without requiring that AxI has a behaviorally closed class of literal algebras. However, we can consider it a real simplification only when $\mathsf{Alg}(\Sigma I, AxI) \subseteq \mathsf{Alg}_{\mathbf{COL}}(\Sigma I_{\mathbf{COL}}, AxI))$, because it reduces the number of models for which the axioms Ax should be proved. The second point, mentioned also in that paper, is that the rule is complete only for behaviorally closed specifications. We will give below a counterexample that shows a correct **COL**-refinement whose correctness cannot be proved using the given rule.

Example 2.45 (Incompleteness of Rule 2.1)**.** *Let $SPI_{\mathbf{COL}} = (\Sigma I_{\mathbf{COL}}, AxI)$ and $SP_{\mathbf{COL}} = (\Sigma_{\mathbf{COL}}, Ax)$ where*

spec $SPI_{\mathbf{COL}}$

sorts s_0, s_1, v

operations $a_0, b_0 : s_0$

$a_1, b_1 : s_1$

$obs_0 : s_0 \to v$

$obs_1 : s_1 \to v$

$f : s_0 \to s_1$

observers obs_0, obs_1

axioms $obs_0(a_0) = obs_0(b_0)$

$f(a_0) = a_1$

$f(b_0) = b_1$

end

spec $SP_{\mathbf{COL}}$

sorts s_1, v

operations $a_1, b_1 : s_1$

$obs_1 : s_1 \to v$

observers obs_1

axioms $a_1 = b_1$

end

Let $k : \mathsf{Alg}(\Sigma I) \to \mathsf{Alg}(\Sigma)$ *be the reduct along the inclusion* $\sigma : \Sigma \to \Sigma I$*. We can easily see that k gives a correct implementation constructor from* $SPI_{\mathbf{COL}}$ *to* $SP_{\mathbf{COL}}$*; for each model* $AI \in SPI_{\mathbf{COL}}$ *the reduct* $AI{\restriction_\sigma}$ *trivially satisfies the* $\Sigma_{\mathbf{COL}}$ *constraints. Because* obs_0 *is the only observer on* s_0*, it means that* $a_0 \approx_{\Sigma I_{\mathbf{COL}}, AI} b_0$ *and furthermore using the fact that f preserves observationally equality we get that* $a_1 \approx_{\Sigma I_{\mathbf{COL}}, AI} b_1$*. This means that* $AI \models obs_1(a_1) = obs_1(b_1)$ *and hence that* $AI{\restriction_\sigma} \models obs_1(a_1) = obs_1(b_1)$ *and finally that* $AI{\restriction_\sigma} \models_{\Sigma_{\mathbf{COL}}} a_1 = b_1$*.*

However, this cannot be proved using the Rule 2.1 as there are standard algebras of AxI that interpret $obs_1(a_1)$ *differently than* $obs_1(b_1)$ *and so the observational equality of* a_1 *and* b_1 *is not a valid property for* $\mathsf{Alg}(\Sigma I, AxI){\restriction_\sigma}$*.*

To sum up, Rule 2.1 fails to accommodate behaviorally non-closed classes in two ways. It is sound for such classes but it is not really a simplification; and also there are cases in which its application for behaviorally non-closed classes, like in Example

2.45, is impossible.

Our solution in order to give a proof rule that is both sound and complete for all basic specifications is to reduce the class of models to the most important ones, i.e. the reachable and fully-abstract models.

$$\frac{k(RFA_{\Sigma I_{\mathbf{COL}}}(\mathsf{Alg}(\Sigma I, AxI))) \models_{\Sigma_{\mathbf{COL}}} Ax}{k(\mathsf{Alg}_{\mathbf{COL}}(\Sigma I_{\mathbf{COL}}, AxI)) \models_{\Sigma_{\mathbf{COL}}} Ax} \tag{2.2}$$

Theorem 2.46 (Soundness and completeness)**.** *Proof rule 2.2 is sound and complete for all* **COL** *constructions.*

Proof. Soundness:

Assume $k(RFA_{\Sigma I_{\mathbf{COL}}}(\mathsf{Alg}(\Sigma I, AxI)) \models_{\Sigma_{\mathbf{COL}}} Ax$, and let $AI \in \mathsf{Alg}_{\mathbf{COL}}(\Sigma I_{\mathbf{COL}}, AxI)$. From Proposition 2.38 we get that there exists $BI \in RFA_{\Sigma I_{\mathbf{COL}}}(\mathsf{Alg}(\Sigma I, AxI)$ such that $AI \equiv_{\Sigma I_{\mathbf{COL}}} BI$. As k preserves **COL**-isomorphisms we have that $k(AI) \equiv_{\Sigma_{\mathbf{COL}}} k(BI)$ and from $k(BI) \models_{\Sigma_{\mathbf{COL}}} Ax$ we conclude, using Corollary 2.22, that $k(AI) \models_{\Sigma_{\mathbf{COL}}} Ax$.

Completeness:

Assume $k(\mathsf{Alg}_{\mathbf{COL}}(\Sigma_{\mathbf{COL}}, AxI)) \models_{\Sigma_{\mathbf{COL}}} Ax$. The conclusion is immediate because $RFA_{\Sigma I_{\mathbf{COL}}}(\mathsf{Alg}(\Sigma I, AxI) \subseteq \mathsf{Alg}_{\mathbf{COL}}(\Sigma_{\mathbf{COL}}, AxI)$. □

Please notice that the additional power of this rule comes from identifying the representative models for a **COL** specification, and exactly as in Definition 2.40 of behavioral consistency, these should be the reachable and fully abstract models. Also, notice that Rule 2.2 not only ensures completeness but it also provides a more restricted class $RFA_{\Sigma I_{\mathbf{COL}}}(\mathsf{Alg}(\Sigma I, AxI))$ than the class $\mathsf{Alg}(\Sigma I, AxI)$ used in Rule 2.1, which gives supplementary power in the process of proving axioms in Ax. In the attempt to prove Ax one can use the knowledge that source algebras are reachable and fully abstract as well as literal models of the source specification.

2.6 Summary

In this chapter we have introduced the basic elements for the theory of algebraic specifications needed in order to investigate the refinement process. In addition to that we have presented a new point of view on some old foundational issues about the behavioral semantics of specifications. It is noted that a more adequate description of behavioral specifications can be given if one chooses carefully the notion of behavioral

consistency, and based on that we improve a proof technique that can be used to ease the assessment of correctness of behavioral refinements.

Chapter 3

Constructions in general contexts

Contents

In this chapter we look at consequences of *stability* [Sch87, BST08, STar] in the context of refinements for behavioral specifications. This investigation is partly motivated by the similarities between stability and *parametricity* [Rey74]. Both stability and parametricity were invented to capture the phenomenon of abstraction barriers: stability in the framework of algebraic specifications; and parametricity in second order lambda calculus, or System F [Gir72, Rey74]. Informally, each of them represents a requirement on functions that manipulate data types, expressed in terms of preservation of relations. Stability is a property of constructions between specifications which requires that relations between source models must be extended to relations between target models. Parametricity is a property of the inhabitants of universally quantified types in System F and requires that type instantiation for related types produces related results. We do not attempt to formalize the relation between these two concepts but rather use the apparent similarities to guide our exploration in the algebraic specifications world so as to mimic older results obtained using parametricity in second order lambda calculus. Something similar to "theorems for free", noticed by Wadler

in [Wad89] for System F, are used to ease correctness proofs of behavioral refinement for algebraic specifications.

3.1 Introduction

Typically, software development is done by splitting a big goal into smaller, more feasible tasks. The small modules are implemented separately and then linked together. We can translate some properties of individual subcomponents to the bigger final product, but we will show that one can also derive new properties based on the mutual independence of the components. Because components are implemented in isolation, and must be ready for use in any global context, i.e. in combination with any implementation of all the other components, they should have a great degree of genericity, i.e. they should not interfere with the properties of other components that are beyond their scope. This kind of reasoning, based on the encapsulation of module implementations captured by stability, will be the subject of this chapter and constitutes the essence of "theorems for free" for refinements.

The development of these ideas starts in Section 3.2 where we introduce the procedure of lifting constructions across fitting signature morphisms and the notion of global constructions. Recall that constructions (Definition 2.31) are basically functions that map the models of a source specification to models of a target specification. We consider that the source and the target signature of a construction represent the local context of its definition. Global constructions are those that are defined in local contexts but can be reused in suitable global contexts. Global contexts are potentially richer than the local contexts, i.e. more operations are present in the signatures. Our goal is to characterize global constructions in terms of preservation of relations and also in terms of definability. Depending on the form of the signature morphisms that map the local contexts to the global ones, we will get different versions of global constructions. In this chapter we look at global constructions that can be reused in absolutely every global context, which will be called general global constructions, and we postpone the investigation of those that are reusable only in some restricted contexts to Chapter 4.

We prove in Section 3.3 that globality, i.e. the capability of a construction to be reused in global contexts, is equivalent to preservation of closed algebraic relations. These relations will be called *closed correspondences*. In Section 3.4 we use that preservation result in order to show that operations implemented via general global constructions, i.e. present in the target signature of such a construction, are definable

in terms of the operations present in the source signature. Finally, Section 3.5 illustrates how the genericity of global constructions can be used to derive properties about such a construction without knowing its actual implementation.

Please have in mind that across the thesis we will use the term genericity (and generic construction) as a synonym to globality (and respectively global construction).

3.2 Local constructions in global contexts

"Local constructions in global contexts" [BST08] is an implementation strategy used for splitting refinement goals into smaller tasks. We will first present the general setting for lifting local constructions to global contexts and then we shall discuss how the special features of **COL** interact fit into the established theory.

Typically a refinement step $SP'_{\mathbf{COL}} \rightsquigarrow SPI'_{\mathbf{COL}}$ means taking models of a more concrete specification (here represented by $SPI'_{\mathbf{COL}}$), and producing models of a more abstract specification (here represented by $SP'_{\mathbf{COL}}$); in other words giving a construction $k' : SPI'_{\mathbf{COL}} \Rightarrow SP'_{\mathbf{COL}}$. Doing that all the time is not only hard, especially when the specified systems have significant details, but it is also inefficient because parts of the construction could perhaps have been reused from a previous implementation. The solution is to isolate parts of $SPI'_{\mathbf{COL}}$, reducing complexity, in terms of which we shall implement parts of $SP'_{\mathbf{COL}}$. Technically this means that we choose a sub-specification $SPI_{\mathbf{COL}}$ of $SPI'_{\mathbf{COL}}$ with less information and use only that information for implementing the corresponding subpart $SP_{\mathbf{COL}}$ of $SP'_{\mathbf{COL}}$.

We will focus on some special **COL**-constructions. These special constructions preserve their arguments and do not enrich the universe of sorts. The first requirement means that we will work only with constructions $k : SPI_{\mathbf{COL}} \Rightarrow SP_{\mathbf{COL}}$ that are persistent along a signature morphism $\sigma_{\mathbf{COL}} : \Sigma I_{\mathbf{COL}} \to \Sigma_{\mathbf{COL}}$ (see Definition 2.32). The second requirement appears because our main focus is to express definability results for new operations in terms of the old operations, and therefore we are not interested in morphisms that enrich the set of sorts but only in those that enrich the set of operations. This will be enforced by assuming that the underlying signature morphisms $\sigma : \Sigma I \to \Sigma$ are tight, i.e. they do not add new sorts. Hence we will work only with tight constructions (see Definition 2.33).

The connection between signatures that appear in a lifting operation can be represented diagrammatically as a pushout in the category of algebraic signatures, which

will be called from now on a *lifting pushout*:

$$\begin{array}{ccc} \Sigma I'_{\mathbf{COL}} & \xrightarrow{\sigma'_{\mathbf{COL}}} & \Sigma'_{\mathbf{COL}} \\ {\scriptstyle\phi}\uparrow & & \uparrow{\scriptstyle\phi'} \\ \Sigma I_{\mathbf{COL}} & \xrightarrow[\sigma_{\mathbf{COL}}]{} & \Sigma_{\mathbf{COL}} \end{array}$$

A particular case of pushout is when $\sigma_{\mathbf{COL}}$ and ϕ are inclusions of common symbols into $\Sigma_{\mathbf{COL}}$ and respectively $\Sigma I'_{\mathbf{COL}}$, and then $\Sigma'_{\mathbf{COL}}$ is the union of $\Sigma_{\mathbf{COL}}$ and $\Sigma I'_{\mathbf{COL}}$.

The goal of this methodology is to get automatically a persistent construction k', whenever a persistent construction k is given, such that the following diagram commutes:

$$\begin{array}{ccc} SPI'_{\mathbf{COL}} & \xRightarrow{k'} & SP'_{\mathbf{COL}} \\ {\scriptstyle\restriction_\phi}\Downarrow & & \Downarrow{\scriptstyle\restriction_{\phi'}} \\ SPI_{\mathbf{COL}} & \xRightarrow[k]{} & SP_{\mathbf{COL}} \end{array}$$

Having such a procedure for lifting constructions means that in order to produce a construction k' in the global context, from $SPI'_{\mathbf{COL}}$ to $SP'_{\mathbf{COL}}$, it is sufficient to focus our work on defining a construction k in the local context and then k' will come automatically. To describe the general lifting procedure we will use the fact that for every lifting pushout we can glue together models over $\Sigma I'$ and Σ, i.e. we use the *amalgamation* property for standard algebras.

Let us recall a well known property of **FOL** (see [Tar86] and [Dia08] for a institutional presentation of this fact): pushout squares of algebraic signatures have the *amalgamation property*:

$$\begin{array}{ccc} \Sigma I' & \xrightarrow{\sigma'} & \Sigma' \\ {\scriptstyle\phi}\uparrow & & \uparrow{\scriptstyle\phi'} \\ \Sigma I & \xrightarrow[\sigma]{} & \Sigma \end{array}$$

This mean that for every $AI' \in \mathsf{Mod}(\Sigma I')$ and $A \in \mathsf{Mod}(\Sigma)$ such that $AI'\restriction_\phi = A\restriction_\sigma$ there exits a unique model $A' \in \mathsf{Mod}(\Sigma')$, denoted by $AI' \otimes A$, such that $A'\restriction_{\sigma'} = AI'$ and $A'\restriction_{\phi'} = A$. Amalgamation is the standard tool for defining liftings and combining specifications in the literature of algebraic specifications (see [Gog89]). Categorically, this condition means that $\mathsf{Mod} : \mathbb{S}ign^{op} \to \mathbb{C}at$ weakly preserves pushouts, i.e. every signature pushout is taken into a weak pullback of model classes.

Before going any further we will make the assumption that all signature morphisms that appear from now on are inclusions. This assumption is made in order to simplify notation only, but all the results should carry over to the non-injective case without

significant effort. Actually, there are plenty of practical examples when one needs non-injective fitting morphisms. Just to give an example imagine a local context that specifies a list of elements of sort $Elem$ and an observer $length : List \to Nat$ that gives the number of elements in a list. It is very plausible that we would want to use constructions for this data type in global context where $Elem = Nat$, hence contexts which are mapped using non-injective fitting morphisms. Technically, we base our simplification assumption on the fact that amalgamated algebras exist disregarding whether the morphisms of the pushout are injective or not.

Now, returning to the lifting procedure notice that we did not state that the fitting morphisms ϕ or ϕ' are **COL** signature morphisms; that is why the lifting pushout is a pushout in the category of standard algebraic signatures rather than in that of **COL**-signatures. We only require that $\sigma_{\mathbf{COL}}$ and $\sigma'_{\mathbf{COL}}$ are **COL**-morphisms. We will use various kinds of fitting morphisms and we will investigate the properties of liftable constructions depending on the properties of the fitting morphisms. The classes of fitting morphisms analyzed in the next chapters (Chapter 4, Chapter 5) will be closed under pushouts along **COL**-signature morphisms, i.e. ϕ' will be in the same class ϕ is. In this chapter we will examine the unrestricted case when the fitting morphism ϕ is an arbitrary signature morphism. This means that global contexts can have any observability and reachability constraints relative to the local contexts, the only requirement being to include the local signatures.

Because ϕ is not necessarily a **COL** signature morphism it follows that in general the reduct along ϕ of a $\Sigma I'_{\mathbf{COL}}$-algebra need not yield a **COL**-algebra over $\Sigma I_{\mathbf{COL}}$. To avoid cluttering conditions about the fitting morphism ϕ we will work in a simplified setting without restricting the usability of the presented technique. More exactly, we will consider only the essential case in which the specifications $SPI'_{\mathbf{COL}}$ and $SP'_{\mathbf{COL}}$ in the global context are the canonical translated specifications obtained from the local context using the morphism ϕ, i.e. $\phi(SPI_{\mathbf{COL}})$ and respectively $\phi'(SP_{\mathbf{COL}})$. This approach is simple enough to ensure that reducts of models from the global specifications are **COL**-algebras but it is also versatile enough to be adapted to global contexts that are more complex. For example, if our global context contains the specifications $SPI'_{\mathbf{COL}}$ and $SP'_{\mathbf{COL}}$, and we decide as before to isolate the specifications $SPI_{\mathbf{COL}}$ and $SP_{\mathbf{COL}}$ as the local context, we distinguish two steps in order to use a construction from the local context. Firstly, we will be able to lift a construction $k : SPI_{\mathbf{COL}} \Rightarrow SP_{\mathbf{COL}}$ to a construction $k' : \phi(SPI_{\mathbf{COL}}) \Rightarrow \phi'(SP_{\mathbf{COL}})$ in a canonical global context. Secondly, we should prove that our original global context fits the canonical one, i.e.

$SPI'_{\mathbf{COL}} \subseteq \phi(SPI_{\mathbf{COL}})$ and $\phi'(SP_{\mathbf{COL}}) \subseteq SP'_{\mathbf{COL}}$, and hence that the canonical lifting is trivially extended to the original global context. However, the second inclusion is quite often impossible to prove, as there is always need to use axioms of $SPI'_{\mathbf{COL}}$ in order to do a proper lifting to a global context. A more appropriate strategy would be to prove $\sigma'_{\mathbf{COL}}(SPI'_{\mathbf{COL}}) \cap \phi(SP_{\mathbf{COL}}) \subseteq SP'_{\mathbf{COL}}$ but even this one fails in some cases. The exact technicalities of the lifting process are depicted in Proposition 3.4. Nevertheless, one can study lifting to the canonical context without caring too much about fitting, which remains to be solved separately.

With this technical simplification our goal changes into that of lifting any construction k between $SPI_{\mathbf{COL}}$ and $SP_{\mathbf{COL}}$ such that the following diagram commutes:

$$\begin{array}{ccc} \phi(SPI_{\mathbf{COL}}) & \xRightarrow{k'} & \phi'(SP_{\mathbf{COL}}) \\ \Big\Downarrow{\scriptstyle \restriction_{\phi}} & & \Big\Downarrow{\scriptstyle \restriction_{\phi'}} \\ SPI_{\mathbf{COL}} & \xRightarrow[k]{} & SP_{\mathbf{COL}} \end{array}$$

In the conventional setting of standard algebras the amalgamability property for the given pushout of signature morphisms is enough to define the lifted version of a construction to a bigger context. However, in **COL** that is no longer the case as there is no guarantee that the standard amalgamation of two **COL**-models is a **COL**-algebra. Even if we were working only with **COL**-signature morphisms, the amalgamation property is not guaranteed for the **COL**-institution as remarked in [BH06a]. Therefore we will define the *standard lifted construction* based on amalgamation for standard algebraic signatures and then we will later see how we can reason about the **COL** constraints.

Definition 3.1 (The standard lifted construction)**.** *Consider a lifting pushout and let* k : $SPI_{\mathbf{COL}} \Rightarrow SP_{\mathbf{COL}}$ *be a persistent* **COL***-construction. The* standard lifted construction *corresponding to* k *is a function* $k' : \phi(SPI_{\mathbf{COL}}) \Rightarrow \mathsf{Mod}(\Sigma')$ *defined as* $k'(AI') = AI' \otimes k(AI'\restriction_{\phi})$. *We will sometimes write* $\phi(k)$ *for the lifted construction* k'.

The question is under what conditions the standard lifted constructions are "well-defined", i.e. they produce **COL**-algebras and preserve **COL**-isomorphisms. We will say that a construction k is *global* if its standard liftings along "all" lifting pushouts are indeed "well-defined". We will parametrize this definition by making it dependent on the class of lifting pushouts that we want to consider. That is in order to give a definition that will form a template for all the notions discussed in this chapter but also in the following ones.

Definition 3.2 (Global **COL**-construction)**.** *Let* $k : SPI_{\mathbf{COL}} \Rightarrow SP_{\mathbf{COL}}$ *be a persistent* **COL***-construction and* $\mathcal{P}$ *be a class of lifting pushouts. We say that* k *is a* $\mathcal{P}$-global **COL**-construction *if for every lifting pushout from* $\mathcal{P}$*,* $\phi(k)$ *is a* **COL***-construction between* $\phi(SPI_{\mathbf{COL}})$ *and* $\phi'(SP_{\mathbf{COL}})$*.*

As we have already announced, in this chapter we will treat the case of constructions that can be lifted across all lifting pushouts, i.e. $\mathcal{P}$ is the class of all lifting pushouts. The adapted version of the previous definition is given below in Definition 3.3. Various instances of the previous definition will be encountered in the next chapters, specialized to the class of signature morphisms used as fitting morphisms, see Definition 4.1, Definition 4.25, Definition 5.11.

Definition 3.3 (General global **COL**-construction)**.** *Let* $k : SPI_{\mathbf{COL}} \Rightarrow SP_{\mathbf{COL}}$ *be a persistent* **COL***-construction. We say that* k *is a* general (global) **COL**-construction *if for every lifting pushout,* $\phi(k)$ *is a* **COL***-construction between* $\phi(SPI_{\mathbf{COL}})$ *and* $\phi'(SP_{\mathbf{COL}})$*.*

Before going any further we will present the typical scenario that involves the reusability of general global constructions. We will explain briefly how one can define a construction in an arbitrary global context starting from the canonical one.

$$\begin{array}{ccc} SPI'_{\mathbf{COL}} & \xRightarrow{k'} & SP'_{\mathbf{COL}} \\ \Big\Uparrow{\scriptstyle \phi} & & \Big\Uparrow{\scriptstyle \phi'} \\ SPI_{\mathbf{COL}} & \xRightarrow[k]{} & SP_{\mathbf{COL}} \end{array}$$

Proposition 3.4 (Lifting general global constructions)**.** *Let* $k : SPI_{\mathbf{COL}} \Rightarrow SP_{\mathbf{COL}}$ *be a general global* **COL***-construction, and two specifications* $SPI'_{\mathbf{COL}}$ *and* $SP'_{\mathbf{COL}}$ *forming a global context connected by the following lifting pushout*

$$\begin{array}{ccc} \mathsf{Sig}[SPI'_{\mathbf{COL}}] & \xrightarrow{\sigma'_{\mathbf{COL}}} & \mathsf{Sig}[SP'_{\mathbf{COL}}] \\ {\scriptstyle \phi}\uparrow & & \uparrow{\scriptstyle \phi'} \\ \mathsf{Sig}[SPI_{\mathbf{COL}}] & \xrightarrow[\sigma_{\mathbf{COL}}]{} & \mathsf{Sig}[SP_{\mathbf{COL}}] \end{array}$$

If the following conditions are satisfied:

1. $SPI'_{\mathbf{COL}} \subseteq \phi(SPI_{\mathbf{COL}})$
2. $\phi(k)(SPI'_{\mathbf{COL}}) \subseteq SP'_{\mathbf{COL}}$

then $k' : SPI'_{\mathbf{COL}} \Rightarrow SP'_{\mathbf{COL}}$*, defined as* $k'(A') = \phi(k)(A')$ *for each* $A' \in \mathsf{Mod}[SPI'_{\mathbf{COL}}]$*, is a* **COL***-construction. Moreover, the following conditions represent an equivalent*

version of the ones above, but they are more appropriate for practical use because they require the inclusion proofs to be done for a smaller class of algebras.

1. $SPI'_{\mathbf{COL}} \subseteq \phi(SPI_{\mathbf{COL}})$
2. $\phi(k)(RFA(SPI'_{\mathbf{COL}})) \subseteq SP'_{\mathbf{COL}}$

Proof. The fact that k' is a **COL**-construction is immediate from the fact that it is a restriction of the canonical lifting $\phi(k)$.

The interesting part are the sufficient conditions that allow us to consider only the reachable and fully-abstract algebras of the input specification from the global context. The simplification is due to the fact that $\phi(k)$ preserves **COL**-isomorphisms, by definition of **COL**-constructions. □

We will now proceed to identify local conditions that ensure the possibility of lifting **COL**-constructions to any general global context.

3.3 General correspondences

In order to obtain **COL**-algebras from lifted constructions $\phi(k)$ for any standard signature morphism ϕ we have to impose additional conditions on the given construction k. The idea is to require k to preserve not only the observational equalities induced by its local context, but all such observational equalities for all possible contexts. As we plan to use k in unrestricted contexts, observational equalities in the global context are not comparable in general to the ones in the local context, i.e. they can be defined for any number of elements and can be finer or coarser than the corresponding one from the local context. Therefore we should require k to be ready to preserve all algebraic relations between $SPI_{\mathbf{COL}}$-algebras.

Definition 3.5 (General correspondence)**.** *Let $\Sigma_{\mathbf{COL}}$ be a* **COL***-signature and A, B be two $\Sigma_{\mathbf{COL}}$-algebras. A* (general) correspondence $\rho\colon A \leftrightarrow B$ *is an algebraic relation. If in addition ρ is a closed relation we call it a* closed general correspondence*. Any two elements $a \in A$ and $b \in B$ such that $a\ \rho\ b$ are called* (general) correspondents*.*

Notice that in contrast to a **COL**-homomorphism (Definition 2.19), a general correspondence is not required in any way to take into account the reachability or observability aspects induced by the (local) **COL**-signature. That happens because it must

be appropriate for all global contexts, i.e. for whatever set of generated elements or observational equality is induced by the global context.

Also, compared with the conventional notion of correspondence invented by Schoett and presented in [Sch87, BST08, STar], we don't require that the relation should be identity, or a bijection, on visible sorts. That is because that we must be ready for fitting morphisms that map visible sorts to loose sorts and therefore we should also consider relations that are not one-to-one correspondences on visible sorts. In Chapter 4 we will give other definitions for some special kinds of correspondence that are closer to the conventional definition.

Now we can express locally, following the pattern already established in [Sch87, BST08], a necessary and sufficient condition for a **COL**-construction to be global. We will show that globality is equivalent to preservation of all closed correspondences. In this case we cannot eliminate the requirement for closed correspondences; it is not true that general global constructions preserve all general correspondences. However, the cases investigated in Chapter 4 will prove to be more adequate and will allow the proof of a result that does not mention closedness.

Definition 3.6 (Generally stable **COL**-construction)**.** *Let* $k : SPI_{\mathbf{COL}} \Rightarrow SP_{\mathbf{COL}}$ *be a tight* **COL***-construction. We say that k is* generally stable *if it extends general correspondences, i.e. for any* $SPI_{\mathbf{COL}}$*-algebras* AI, BI *and any general correspondence* $\rho: AI \leftrightarrow BI$ *we have that* ρ *is a general correspondence between* $k(AI)$ *and* $k(BI)$*. We say that k is* generally closed-stable *if it extends closed general correspondences.*

Please notice that the definition of stable constructions as we gave it relies on the fact that the construction is tight, because it makes sense saying that a ΣI-relation can be seen directly as a Σ-relation because there are no sorts in Σ that are not also in ΣI.

We can now show that the requirement of general closed-stability is necessary and sufficient for a tight **COL**-construction to be usable in all general contexts. Again, our result is different than previous results obtained in the literature, as globality is proven to be equivalent to closed-stability rather than to plain stability. That happens because the general case was not previously investigated and the characterization of globality in terms of plain stability is possible only in the special cases presented in Chapter 4.

Proposition 3.7 (Closed-stability implies globality)**.** *Let* $k : SPI_{\mathbf{COL}} \Rightarrow SP_{\mathbf{COL}}$ *be a tight* **COL***-construction. If k is generally closed-stable then k is a general global* **COL***-construction.*

Proof. Consider a lifting pushout as below:

$$\begin{array}{ccc} \Sigma I'_{\mathbf{COL}} & \xrightarrow{\sigma'_{\mathbf{COL}}} & \Sigma'_{\mathbf{COL}} \\ {\scriptstyle\phi}\uparrow & & \uparrow{\scriptstyle\phi'} \\ \Sigma I_{\mathbf{COL}} & \xrightarrow[\sigma_{\mathbf{COL}}]{} & \Sigma_{\mathbf{COL}} \end{array}$$

We need to show that $\phi(k)$ is a persistent **COL**-construction along $\sigma'_{\mathbf{COL}}$ between $\phi(SPI_{\mathbf{COL}})$ and $\phi'(SP_{\mathbf{COL}})$. For simplicity we will write k' instead of $\phi(k)$.

First of all we will show that $k'(AI') \in \mathsf{Alg}_{\mathbf{COL}}(\Sigma'_{\mathbf{COL}})$ for all $AI' \in \mathsf{Alg}_{\mathbf{COL}}(\Sigma I'_{\mathbf{COL}})$. Let $AI = AI'\restriction_{\phi}$, $A = k(AI)$ and $A' = k'(AI')$.

An important observation is that the **COL**-constraints induced by the global signatures on AI' and A' coincide, and this happens also for the local context. This amounts to proving that $\langle Gen_{\Sigma I'_{\mathbf{COL}}}(AI')\rangle = \langle Gen_{\Sigma'_{\mathbf{COL}}}(A')\rangle\restriction_{\sigma'}$, $\langle Gen_{\Sigma I_{\mathbf{COL}}}(AI)\rangle = \langle Gen_{\Sigma_{\mathbf{COL}}}(A)\rangle\restriction_{\sigma}$, $\approx_{\Sigma I'_{\mathbf{COL}},AI'} = \approx_{\Sigma'_{\mathbf{COL}},A'}\restriction_{\sigma'}$ and $\approx_{\Sigma I_{\mathbf{COL}},AI} = \approx_{\Sigma_{\mathbf{COL}},A}\restriction_{\sigma}$. We will do the proof only for the global context as the claims for the local context are a particular case of the global one. Because $\sigma'_{\mathbf{COL}}$ is a **COL** signature morphism we trivially get that observational equalities and generated parts induced by the source and the target signatures coincide. Also, we get that $\langle Gen_{\Sigma I'_{\mathbf{COL}}}(AI')\rangle \subseteq \langle Gen_{\Sigma'_{\mathbf{COL}}}(A')\rangle\restriction_{\sigma'}$. In order to prove that these generated algebras are equal we will show that the newly added operations do not produce new elements. For that let $\approx_I$ be a relation on AI that is equal to the restriction $\approx_{\Sigma I'_{\mathbf{COL}},AI'}\restriction_{\phi}$. It is clear that $\approx_I$ is a correspondence that is bi-surjective on $\langle Gen_{\Sigma I'_{\mathbf{COL}}}(AI')\rangle\restriction_{\phi}$. Furthermore, $\approx_I$ is closed because it is the restriction of an observational equality. Because k is generally closed stable we get that $\approx_I$ is also a correspondence on A. Hence, the newly added operations preserve $\approx_I$, and therefore when they are applied to elements from $\langle Gen_{\Sigma I'_{\mathbf{COL}}}(AI')\rangle\restriction_{\phi}$ they produce elements from $\langle Gen_{\Sigma I'_{\mathbf{COL}}}(AI')\rangle\restriction_{\phi}$. This ensures that $\langle Gen_{\Sigma'_{\mathbf{COL}}}(A')\rangle\restriction_{\sigma'} \subseteq \langle Gen_{\Sigma I'_{\mathbf{COL}}}(AI')\rangle$ due to the minimality of the generated algebras.

To show that A' is a **COL** algebra we need to show that $\approx_{\Sigma'_{\mathbf{COL}},A'}$ is a Σ'-congruence on $\langle Gen_{\Sigma'_{\mathbf{COL}}}(A')\rangle$. For the operations that come from $\Sigma I'$ the preservation requirement is immediate. We only have to prove it for operation that are new. Such an operation $op' \in OP'$ comes from Σ and hence it preserves $\approx_I$.

Secondly we need to show that for all $AI' \equiv_{\Sigma I'_{\mathbf{COL}}} BI'$ we have $k(AI') \equiv_{\Sigma'_{\mathbf{COL}}} k(BI')$. Let h be the isomorphism between AI' and BI' and ρ be the closed correspondence equal to $h\restriction_{\phi}$. From the fact that k is generally closed stable we get that ρ is preserved by the implemented operations. Furthermore, because σ' is tight h will be an isomorphism also between $k(AI')$ and $k(BI')$. □

In the proof of necessity we will take a closed general correspondence ρ and define a context in which ρ will become a **COL**-isomorphism. Then using globality we will get that such an isomorphism must be preserved.

Proposition 3.8 (Globality implies closed-stability)**.** *Let $k : SPI_{\mathbf{COL}} \Rightarrow SP_{\mathbf{COL}}$ be a tight* **COL***-construction. If k is a general global construction then k is generally closed-stable.*

Proof. Let AI and BI be two $SPI_{\mathbf{COL}}$ models and ρ be a general closed correspondence between AI and BI. We want to prove that ρ is also a closed correspondence between $k(AI)$ and $k(BI)$. It is sufficient to prove that the additional operations introduced in Σ commute with ρ (closedness is immediate because it does not depend on the signature). For that we will build a global context $\Sigma I'_{\mathbf{COL}}$ and two $\Sigma I'_{\mathbf{COL}}$-isomorphic extensions AI' and BI' of AI and resp. BI such that the corresponding isomorphism extends ρ and using the fact that the lifting of k to this context must be a **COL**-construction we will infer that k preserves ρ.

Now, let $\Sigma I'_{\mathbf{COL}}$ be the following **COL**-signature $(SI', OPI', OPI'_{Cons}, OPI'_{Obs})$ where

- $SI' = SI \uplus \{Bool\}$
- $OPI' = OPI \uplus \{!^{a,b} : s \mid (a,b) \in \rho_s, s \in SI\} \uplus \{?^b : s \to Bool \mid b \in B_s, s \in SI\} \uplus \{true, false : Bool\}$
- $OPI'_{Cons} = \{!^{a,b} : s \mid (a,b) \in \rho_s, s \in SI\} \uplus \{true, false : Bool\}$
- $OPI'_{Obs} = \{(op, i) \mid op : s_1, \ldots, s_n \to s \in OPI, 1 \leq i \leq n\} \uplus \{?^b : s \to Bool \mid b \in B_s, s \in SI\}$
- $SI'_{Cons} = SI$

Please notice that we have defined explicitly the set of constructed sorts and recall Remark 2.11 for more information about that. A consequence of the way we chose to define $\Sigma I'_{\mathbf{COL}}$ is that all sorts from SI are hidden and constructed. Furthermore, because we have constructors only on sorts for which $dom(\rho)$ is not empty, the generated parts of $\Sigma I'_{\mathbf{COL}}$-algebras are empty on all the other sorts.

Consider the obvious inclusion from ΣI to $\Sigma I'$ to be the fitting morphism ϕ and let us define AI' such that $AI' \restriction_\phi = AI$ and BI' such that $BI' \restriction_\phi = BI$. For that we must define only the interpretation of the additional symbols. We let the sort $Bool$ be interpreted as follows : $AI'_{Bool} = BI'_{Bool} = \{true, false\}$ and for the additional operations

AI': • $!^{a,b} = a$ for all $(a,b) \in \rho$

• $?^b(a) = \mathit{true}$ if $a\ \rho\ b$ and false otherwise

BI': • $!^{a,b} = b$ for all $(a,b) \in \rho$

• $?^b(b') = \mathit{true}$ if there exists a such that $a\ \rho\ b$ and $a\ \rho\ b'$ and false otherwise

Notice that both algebras AI' and BI' satisfy the reachability constraints imposed by $\Sigma I'_{\mathbf{COL}}$ because their reachable parts consist of the elements that are linked by ρ, i.e. $\langle Gen_{\Sigma I'_{\mathbf{COL}}}(AI')\rangle = dom_{AI'}(\rho)$ and $\langle Gen_{\Sigma I'_{\mathbf{COL}}}(BI')\rangle = dom_{BI'}(\rho)$. Also, they satisfy the observability constraint because all operations in the signature are observers and they preserve the observational equality by default.

Now, we will show that the extension of ρ that is identity on *Bool*, called ρ', is a $\Sigma I'_{\mathbf{COL}}$ isomorphism between AI' and BI' by checking all conditions of Definition 2.19 for all sorts $s \in SI$.

- For all $a \in \langle Gen_{\Sigma I'_{\mathbf{COL}}}(AI')\rangle_s$ we get that there exists $b \in BI_s$ such that $a\ \rho_s\ b$ and therefore $b \in \langle Gen_{\Sigma I'_{\mathbf{COL}}}(BI')\rangle_s$.

- For all $a, a' \in \langle Gen_{\Sigma I'_{\mathbf{COL}}}(AI')\rangle_s$ and $b \in \langle Gen_{\Sigma I'_{\mathbf{COL}}}(BI')\rangle_s$ such that $a \approx_s a'$ and $a\ \rho_s\ b$ we have that $?^b(a) = ?^b(a') = \mathit{true}$ and hence $a'\ \rho_s\ b$.

- For all b and $a_0\ \rho_s\ b_0$ we show that $?^b(a_0) = ?^b(b_0)$. If $?^b(a_0) = \mathit{true}$ then $a_0\ \rho_s\ b$ by definition and hence $?^b(b_0) = \mathit{true}$. If $?^b(b_0) = \mathit{true}$ we get that there exists a such that $a\ \rho_s\ b_0$ and $a\ \rho_s\ b$ and then because ρ is closed we get that $a_0\ \rho_s\ b$ and hence $?^b(a_0) = \mathit{true}$.

- For all $a \in \langle Gen_{\Sigma I'_{\mathbf{COL}}}(AI')\rangle_s$ and $b, b' \in \langle Gen_{\Sigma I'_{\mathbf{COL}}}(BI')\rangle_s$ such that $a\ \rho_s\ b$ and $b \approx_s b'$ we have that $?^{b'}(a) = ?^{b'}(b) = ?^{b'}(b') = \mathit{true}$ and hence $a\ \rho_s\ b'$.

- For all $a \in \langle Gen_{\Sigma I'_{\mathbf{COL}}}(AI')\rangle_s$ and $b, b' \in \langle Gen_{\Sigma I'_{\mathbf{COL}}}(BI')\rangle_s$ such that $a\ \rho_s\ b$ and $a\ \rho_s\ b'$ we have that $?^{b_0}(a) = ?^{b_0}(b) = ?^{b_0}(b')$ for all $b_0 \in BI'_s$ and hence $b \approx_s b'$.

- All the operations from *OPI* preserve ρ because it is a correspondence.

We have established that ρ' is an isomorphism between AI' and BI'. Moreover, we can show that any **COL**-morphism h' between AI' and BI' must be equal to ρ'. To see that consider such a **COL**-morphism $h' : AI' \to BI'$. Because the interpretations of the constructors must be related we get easily that $\rho' \subseteq h'$. Now consider $a \in \langle Gen_{\Sigma I'_{\mathbf{COL}}}(AI')\rangle$ and $b \in \langle Gen_{\Sigma I'_{\mathbf{COL}}}(BI')\rangle$ such that $a\ h'\ b$. Because $\langle Gen_{\Sigma I'_{\mathbf{COL}}}(BI')\rangle =$

$dom_{BI'}(\rho)$ we get that there exists a_0 such that $a_0\ h\ b$. But from the properties of the **COL**-morphisms that entails $a \approx_{\Sigma I'_{\mathbf{COL}}, AI'} a_0$ and eventually that $?^b(a) = ?^b(a_0) = true$, which implies $a\ \rho\ b$. That proves $h' \subseteq \rho'$ and finally that $h' = \rho'$.

Using the fact that k is a global construction we get that $k'(AI) \equiv_{\Sigma'_{\mathbf{COL}}} k'(BI)$. But the reduct of the **COL**-isomorphism between $k'(AI')$ and $k'(BI')$ must be the unique **COL**-morphism ρ' between AI and BI. Therefore we get that the additional operations added by σ preserve the relation ρ. □

The equivalence of the two notions, globality and closed-stability, is synthesized in the following theorem.

Theorem 3.9 (Globality is equivalent to closed-stability)**.** *Let $k : SPI_{\mathbf{COL}} \Rightarrow SP_{\mathbf{COL}}$ be a tight* **COL***-construction. Then k is general global construction if and only it is general closed-stable.*

Proof. The two implications are proved in Proposition 3.7 and Proposition 3.8. □

3.4 Definability

This section sets the framework for obtaining useful properties of constructed models from the fact that general global constructions are generally closed-stable. Informally, we can reason that in order to use a construction in any global context, its definition must rely only on the operations provided by the local source specification. Any supplementary assumptions about the source models could be contradicted by some choice of context, so the implementer should write implementations in terms of the available signature. Before formalizing the notion of definability and proving that closed-stability implies it, we will make a small detour by looking at a similar phenomenon that happens in System F.

3.4.1 Definability via parametricity

This section is not crucial for understanding the results presented in this thesis, but it helps to illustrate the intuition behind them: stability could be used for algebraic specifications similarly to how parametricity is used in second order lambda calculus.

The second order lambda calculus was first developed by Girard [Gir72] in the context of logic and was reinvented independently a few years later by Reynolds [Rey74] in the context of computer science as a framework for polymorphic programming

languages. Reynolds introduced the notion of *relational parametricity* [Rey83], and proved an *abstraction theorem* in order to explain semantically the difference between parametric polymorphism and ad-hoc polymorphism. A polymorphic function is parametric, in the sense of Strachey [Str67], if it always uses the same algorithm, regardless of the type it is applied to. This means that a parametric function does not make additional assumptions, it uses only the information available in its polymorphic type.

As a consequence of this requirement for generality, the number of parametric functions of a given polymorphic type can be quite restricted. Reynolds used relations to formalize this principle of generality, and this finally led to the discovery of properties of parametric functions that are derivable directly from their type, called by Wadler *theorems for free* [Wad89]. Rather than giving a definition for System F we will just recall the basic constructs with some examples. In System F we can write universally quantified types like $\forall X.X \times X \to X$. An inhabitant of this type $f : \forall X.X \times X \to X$ can be instantiated for any type A, obtaining a term $f_A : A \times A \to A$. Notice that, using an informal semantics for System F, an inhabitant $f : \forall X.X \times X \to X$ can be considered as an element $f \in \prod_{A \in Types}(A \times A \to A)$, and therefore nothing forces uniformity of interpretation across various types. Examples of inhabitants for this type are the first projection $\lambda x, y{:}X.x$ and the second projection $\lambda x, y{:}X.y$ which are uniform, but also non-uniform polymorphic functions like $\lambda x, y{:}X.if\ (X = Nat)\ then\ x+1\ else\ x$ which acts differently for natural numbers than for other types.

Relational parametricity [Rey83] is a semantic constraint that distinguishes between the uniform, so-called parametric, and the non-uniform inhabitants of types by requiring that whenever we instantiate an inhabitant of a universal type to related types we obtain related results. In order to make this formal it is necessary to define the action of type constructors on relations. For products, two pairs are related if the corresponding components are related; for functions, two functions are related if they take related arguments to related results. In our case $f : \forall X.X \times X \to X$ is parametric if for every relation $R \subseteq A \times B$, $f_A(R \times R \to R)f_B$. Now, it is easy to prove that the only parametric inhabitants of the type $\forall X.X \times X \to X$ are the two projections. This can be done as follows. Let $f : \forall X.X \times X \to X$ be a parametric function, and assume that $f_{Bool}(true, false) = true$. Now, consider a type A and two elements of this type $a, b : A$; we will show $f_A(a,b) = a$. For this we take a relation $R \subseteq Bool \times A$ such that $R = \{(true, a), (false, b)\}$ and due to parametricity we obtain $f_{Bool}(true, false)\ R\ f_A(a,b)$, i.e. $f_A(a,b) = a$. If $f_{Bool}(true, false) = false$ we can also show that $f_A(a,b) = b$ for every $a, b : A$. Therefore, f acts like the first argument

projection.

There are many known consequences of parametricity, especially definability results based on the representability of initial algebras in System F [RP90, GLT89, PA93]. For example: there is only one parametric inhabitant of the type $\forall X.X \to X$, namely the polymorphic identity function. Another classic example is that the only parametric inhabitants of the type $\forall X.(X \to X) \times X \to X$ are the definable terms $\lambda s : X \to X.\lambda 0 : X.s^n(0)$, the Church numerals.

Even though we will not look for a formal translation of these results into the setting of algebraic specifications, we will investigate the idea that stable constructions should be definable. In other words, we are trying to draw an informal parralel between stability and parametricity and obtain for the former the same kind of results that have already been obtained for the later.

3.4.2 Definability via stability

In order to prove that stability implies definability we shall define what we mean for an operation interpreted in an algebra to be definable in terms of the operations of another signature.

Definition 3.10 (Algebraic definability)**.** *Let Σ and Σ' be two algebraic signatures and let $\sigma : \Sigma \to \Sigma'$ be a tight signature morphism between them. Let $op' : s_1, \ldots, s_n \to s \in OP'$ be an operation symbol and A' be a Σ'-algebra.*

We say that op' is (point-wise) Σ-definable *on A' if for every tuple $a_1, \ldots, a_n$, with $a_i \in A'_{s_i}$ for $i = 1 \ldots n$, there exists a term $t \in T_\Sigma(\{x_1, \ldots, x_n\})$ such that:*

$$A'_{op'}(a_1, \ldots, a_n) = A'_t[a_1/x_1, \ldots, a_n/x_n]$$

We call the term t the defining term *for op' and $a_1, \ldots, a_n$.*

Let $k : SPI_{\mathbf{COL}} \Rightarrow SP_{\mathbf{COL}}$ be a tight **COL***-construction. We say that k is* definable *if all operations from OP are ΣI-definable on algebras in $k(SPI_{\mathbf{COL}})$.*

We can now prove that any generally closed-stable construction must be definable.

Proposition 3.11 (Closed-stability implies definability)**.** *Let $k : SPI_{\mathbf{COL}} \Rightarrow SP_{\mathbf{COL}}$ be a tight* **COL***-construction. If k is generally closed-stable then k is definable.*

Proof. Let A be a **COL**-algebra in $k(SPI_{\mathbf{COL}})$, i.e. there exists $AI \in SPI_{\mathbf{COL}}$ such that $k(AI) = A$. Let $op : w \to s \in OP$ and consider a tuple $\underline{a} \in A_w$. We define a closed

correspondence ρ: $AI \leftrightarrow AI$ that relates $AI_t[\underline{a}]\ \rho\ AI_t[\underline{a}]$ for all $t \in T(\{\underline{x}\})$. Because k extends closed correspondences we get that $AI_{op}(\underline{a})\ \rho\ AI_{op}(\underline{a})$ and hence, from the definition of ρ, we get that there exists a term $t \in T(\{\underline{x}\})$ such that $AI_{op}(\underline{a}) = AI_t[\underline{a}]$. □

A stronger notion of definability is *uniform definability* which requires that the defining terms should be the same for correspondent arguments. We can prove that this concept is equivalent to stability, however it is not equivalent to globality.

Definition 3.12 (Uniform (general) definability)**.** *Let* $\Sigma_{\mathbf{COL}}$ *and* $\Sigma'_{\mathbf{COL}}$ *be two* **COL***-signatures and let* $\sigma_{\mathbf{COL}} : \Sigma_{\mathbf{COL}} \to \Sigma'_{\mathbf{COL}}$ *be a tight* **COL***-signature morphism. We say that an operation* $op' : w \to s \in OP'$ *is* uniform (generally) definable *on a class* $\mathcal{A}'$ *of* $\Sigma'_{\mathbf{COL}}$*-algebras if for all* $A', B' \in \mathcal{A}'$*, all general closed correspondences* ρ: $A'\restriction_\sigma \leftrightarrow B'\restriction_\sigma$*, and all* ρ*-correspondent tuples* $\underline{a}\ \rho\ \underline{b}$ *there exists a common defining term* t *for* op' *and* $\underline{a}$*, resp. for* op' *and* $\underline{b}$*.*

Let $k : SPI_{\mathbf{COL}} \Rightarrow SP_{\mathbf{COL}}$ *be a tight* **COL***-construction. We say that* k *is* uniform generally definable *if all operations from OP are uniform generally definable on* $k(SPI_{\mathbf{COL}})$.

It is easy to prove that uniform general definability is a sufficient condition for obtaining closed-stability.

Proposition 3.13 (Uniform definability implies closed-stability)**.** *Let* $k : SPI_{\mathbf{COL}} \Rightarrow SP_{\mathbf{COL}}$ *be a tight* **COL***-construction. If* k *is uniform generally definable then* k *is generally closed-stable.*

Proof. Let AI, BI be two $\Sigma I_{\mathbf{COL}}$-algebras and ρ: $AI \leftrightarrow BI$ be a general closed correspondence between them. We will denote by A and B the constructed algebras $k(AI)$ and $k(BI)$. Also, consider an operation $op : w \to s \in OP$ and two tuples $\underline{a} \in A_w$ and $\underline{b} \in B_w$ such that $\underline{a}\ \rho_w\ \underline{b}$. Using the general definability assumption we get that there is a defining term t for op and $\underline{a}$ that acts as a defining term also for op and $\underline{b}$, i.e. $A_{op}(\underline{a}) = A_t[\underline{a}]$ and $B_{op}(\underline{b}) = B_t[\underline{b}]$. Because ρ is a ΣI correspondence we get that $A_t[\underline{a}]\ \rho\ B_t[\underline{b}]$ and hence that the interpretations of op in A and B preserve the relation ρ. As that happens for every operation and related arguments, we get that ρ is a Σ-correspondence and therefore k extends closed correspondences. □

We can also prove that stability is sufficient for uniform-definability.

Proposition 3.14 (Stability implies uniform definability)**.** *Let* $k : SPI_{\mathbf{COL}} \Rightarrow SP_{\mathbf{COL}}$ *be a tight* **COL***-construction. If* k *is generally stable then* k *is generally uniform definable.*

Proof. In order to obtain the same defining term for two correspondent arguments we use the term generated correspondence over those arguments. □

The reason why closed-stability is not capable of enforcing uniform definability is because the term generated relations are typically not closed and hence they are not automatically preserved by a closed-stable construction.

So, uniform definability is equivalent to stability but is only sufficient for closed-stability. That also means that uniform definability is sufficient for proving globality. However, the reverse is not true, i.e. there are global constructions for which the implemented operations fail to be definable by the same term for correspondent arguments as we can see in the following example.

Example 3.15 (Closed-stability does not imply uniform definability)**.** *We will define a tight construction* $k : SPI_{\mathbf{COL}} \Rightarrow SP_{\mathbf{COL}}$ *that is generally closed stable but is not uniform definable. For that let* $SPI_{\mathbf{COL}}$ *be the following specification:*

spec $SPI_{\mathbf{COL}}$
  sorts s, v
  operations $out : s \to v$
    $in : v \to s$
    $\bot, \top : v$
  constructors $\bot, \top, in$
  observers out
  axioms $\bot \neq \top$
    $out(in(\bot)) = \bot$
end

and let $SP_{\mathbf{COL}}$ *be the extension of* $SPI_{\mathbf{COL}}$ *with an operation*

spec $SP_{\mathbf{COL}} = SPI_{\mathbf{COL}}$ ***then***
  operations $op : s \to s$
end

A construction $k : SPI_{\mathbf{COL}} \Rightarrow SP_{\mathbf{COL}}$ *must implement op for each algebra* $AI \in SPI_{\mathbf{COL}}$*. Let AI be a model of the specification and let* $A = k(AI)$ *be defined as follows:*

$$A_{op}(x) = A_{in(\top)} \text{ for all } x \in A_s \text{ if } AI_{out(in(\top))} = AI_{\bot}$$

$$A_{op}(x) = A_{in(\bot)} \text{ for all } x \in A_s \text{ if } AI_{out(in(\top))} = AI_{\top}$$

To prove that k is generally closed-stable we must prove that it extends all closed general correspondences. For that we will analyze the pattern of correspondences between algebras of $SPI_{\mathbf{COL}}$ *specification.*

Let $AI, BI \in SPI_{\mathbf{COL}}$ *and* ρ*:* $AI \leftrightarrow BI$ *be a closed general correspondence. Knowing just this information we can deduce that* $AI_{\bot}\ \rho\ BI_{\bot}$*,* $AI_{\top}\ \rho\ BI_{\top}$*,* $AI_{in(\bot)}\ \rho\ BI_{in(\bot)}$ *and* $AI_{in(\top)}\ \rho\ BI_{in(\top)}$*. To further enrich our knowledge about how* ρ *looks we need to assume something about the interpretation of out in the two algebras. We will treat two cases depending on whether the term* $out(in(\top))$ *is interpreted identically in AI and BI or is interpreted differently. First, let us look at the simple case and assume that* $out(in(\top))$ *is interpreted as* $\top$ *or as* $\bot$ *in both models. Then the definition of the construction is the same for AI and BI and without any additional information about* ρ *we can conclude that k extends* ρ*. Secondly, we should look at the case when the construction defines the implementation differently and for that assume that* $AI_{out(in(\top))} = AI_{\bot}$ *and* $BI_{out(in(\top))} = BI_{\top}$*. This allows us to deduce that* $AI_{\bot}\ \rho\ BI_{\top}$ *and because* ρ *is closed we get* $AI_{\top}\ \rho\ BI_{\bot}$ *and furthermore that* $A_{in(\top)}\ \rho\ B_{in(\bot)}$*. The last relation is sufficient to conclude that the interpretations of op preserve* ρ *as* $A_{op}(x) = A_{in(\top)}$ *and* $B_{op}(x) = B_{in(\bot)}$*.*

Finally, we should give an example of two correspondent elements for which the operation op cannot be defined by the same term. Let AI and BI be defined as follows:

$$AI_s = \{a_0, a_1\}\ ,\ BI_s = \{b_0, b_1\}\ AI_v = BI_v = \{\bot, \top\}$$

$$AI_{\bot} = BI_{\bot} = \bot,\ AI_{\top} = BI_{\top} = \top$$

$$AI_{in(\bot)} = a_0\ ,\ AI_{in(\top)} = a_1$$

$$BI_{in(\bot)} = b_0\ ,\ BI_{in(\top)} = b_1$$

$$AI_{out(a_0)} = \bot\ ,\ AI_{out(a_1)} = \bot$$

$$BI_{out(b_0)} = \bot\ ,\ BI_{out(b_1)} = \top$$

and let ρ *be the total relation between AI and BI (obviously* ρ *is a closed general correspondence). From the definition of k we get that* $A_{op}(x) = A_{in(\top)} = a_1$ *and* $B_{op}(x) = B_{in(\bot)} = b_0$*. Now, taking our correspondents via* ρ *to be* a_0 *and* b_0 *we get that there is no term t such that* $A_t[a_0] = a_1$ *and* $B_t[b_0] = b_0$*. Before ending this example please notice that there are particular terms* t_0 *and* t_1 *such that* $A_{t_0} = a_1$ *and* $B_{t_1} = b_0$*, so the result of point-wise definability is not under scrutiny here.*

This example also shows, indirectly, that there exists general global constructions that do not preserve all correspondences; they only preserve the closed ones. That happens because the construction in Example 3.15 cannot preserve all correspondences. Assuming that it does then it should also be uniformly definable which is obviously a contradiction according to what we have just proven.

3.5 Theorems for free

We have shown, in the previous section, that operations implemented via general global constructions are required to be syntactically definable in terms of the source signature. We can use that insight to show that if operations from the source specification preserve a property then the implemented operations, which are definable in terms of the source operations, will preserve the same property.

Our example will be that of implementing a *replace* function for lists in a global context based on the specification LISTWITHNAME starting from a general global construction k acting in a local context based on specification LISTSKELETON.

The specifications defined below are linked by the following lifting pushout

$$\begin{array}{ccc} \mathsf{Sig}[\textsc{ListWithName}] & \longrightarrow & \mathsf{Sig}[\textsc{ListWithNameAndReplace}] \\ \uparrow{\scriptstyle \phi} & & \uparrow{\scriptstyle \phi'} \\ \mathsf{Sig}[\textsc{ListSkeleton}] & \longrightarrow & \mathsf{Sig}[\textsc{ListSkeletonWithReplace}] \end{array}$$

The specifications LISTWITHNAME and LISTWITHNAMEANDREPLACE constitute the global context:

spec LISTWITHNAME
sorts $List, Elem, String$
operations $empty : List$
$insert, remove : Elem, List \rightarrow List$
$change : String, List \rightarrow List$
$name : List \rightarrow String$
constructors $empty, insert, change$
observers $(name)$
axioms $name(insert(x,c)) = name(c)$
$name(remove(x,c)) = name(c)$
$name(change(s,c)) = s$
end

spec LISTWITHNAMEANDREPLACE = LISTWITHNAME **THEN**
operations $replace : Elem, Elem, List \rightarrow List$
axioms **% This is a theorem for free**
$name(replace(x,y,c)) = name(c)$
end

The local context is formed from specifications LISTSKELETON and LISTSKELETONWITHREPLACE.

spec LISTSKELETON
sorts $List, Elem$
operations $insert, remove : Elem, List \rightarrow List$
end

spec LISTSKELETONWITHREPLACE = LISTSKELETON **THEN**
operations $replace : Elem, Elem, List \rightarrow List$
end

We will enumerate some of the differences between the local and the global con-

text. First of all, in the local context there are no hidden or constructed sorts, because *List* and *Elem* are both visible and loose. However, in the global context *List* becomes a hidden and constructed sort. Please notice that the fitting morphism ϕ is not a **COL** signature morphism, moreover it is not even vertical.

So, assume we have a general global construction that implements *replace* locally:

$$k : \text{LISTSKELETON} \Rightarrow \text{LISTSKELETONWITHREPLACE}$$

It follows that we can use the canonical lifting

$$\phi(k) : \phi(\text{LISTSKELETON}) \Rightarrow \phi(\text{LISTSKELETONWITHREPLACE})$$

and by using Proposition 3.4 we can define a construction in the global context

$$k' : \text{LISTWITHNAME} \Rightarrow \text{LISTWITHNAMEANDREPLACE}$$

if we can prove the following two conditions:

- $\text{LISTWITHNAME} \subseteq \phi(\text{LISTSKELETON})$
- $\phi(k)(RFA(\text{LISTWITHNAME})) \subseteq \text{LISTWITHNAMEANDREPLACE}$

The first condition is trivially satisfied, because the local context does not have any **COL**-requirements, i.e. all standard algebras are **COL**-algebras.

The second condition is where genericity of k comes into play in order to derive the desired property $\forall x, y, c.name(replace(x, y, c)) = name(c)$ in the global context. Please notice that we do not know much about k; we do not know exactly how it is implemented but we know it must be reusable in all general contexts. Therefore, k must be definable (see Propostion 3.11), and hence *replace* must be definable in terms of *insert* and *remove*. So, for algebras in $\phi(RFA(\text{LISTSKELETON}))$ for which *insert* and *remove* preserve names it follows easily that *remove* also preserves names.

The example shows that proving the correctness of a lifted construction based on the canonical lifting cannot be done solely using the information of the specifications involved in the lifting procedure, but must rely on the genericity of the construction. That use of genericity to derive properties directly from the type of the k resembles the phenomenon in System F known as "theorems for free". The variant adapted to System F of our "theorem for free" says that every parametric inhabitant (recall that parametricity corresponds to stability) of a type corresponding to the "type" of k:

$$\forall E.\forall L.(E \to L \to L) \to (E \to L \to L) \to (E \to E \to L \to L)$$

can be written as

$$\lambda E, L.\lambda insert, remove.\lambda x, y, c.op(\ldots op(x, y, c)\ldots)$$

where each occurrence of *op* can be replaced either with *insert* or with *remove*. This "theorem for free" in System F basically means that all parametric inhabitants of that type are lambda definable.

3.6 Summary

In this chapter we have presented the adaptation, to the setting of constructor based observational logic [BH06a], of a technique developed in [BST08] for lifting constructions defined locally to global contexts. The kind of construction that we have studied is the most general of all that we will consider in this thesis, because such a construction can be reused in all global contexts.

Global contexts allow the exploration of the constructed models in full detail and they can refer to any number of elements from the carriers and observe them at any granularity. We proved that general globality is equivalent to preservation of closed algebraic relations. This characterization result is inspired from what was first done in [Sch87], but the setting and the final meaning of the result are different. In the next chapter we will see versions of globality that are closer to the ones that have been investigated in the literature. The novelty of our approach consists exactly in the tuning of various notions of relation that are needed to characterize different notions of globality, that vary with the restrictions imposed on their usage contexts.

Another novelty with respect to the algebraic specification literature is our investigation of definability results implied by stability. This research line was suggested by similar results obtained for the second order lambda calculus from parametricity. The definability result obtained for global constructions represents a confirmation of the informal expectation that the implementer should rely only on what it is available to him in the source signature when writing a generally reusable implementation. In the example given in Section 3.5 we have exploited definability in order to show that globally implemented operations inherit properties from the operations present in the local source signature, and one can simply guess such properties by looking at the type of the implementation, as was done by Wadler for System F [Wad89].

Chapter 4

Constructions in restricted contexts

Contents

In this chapter we will study implementation constructions that are prepared to be lifted only to contexts fitted via a restricted class of signature morphisms, in contrast to Chapter 3 where the fitting morphism was allowed to be an arbitrary signature morphism.

4.1 Introduction

In Chapter 3 we have investigated the properties of constructions that can be reused in general contexts. A step further can be made by considering fitting signature morphisms that are more adequate to the behavioral aspects of **COL**. The first case we will look at is that of constructions that can be lifted only via *vertical* pushouts, i.e. those in which ϕ is a vertical signature morphism:

$$\begin{array}{ccc} \Sigma I'_{\mathbf{COL}} & \xrightarrow{\sigma'_{\mathbf{COL}}} & \Sigma'_{\mathbf{COL}} \\ \uparrow{\scriptstyle \phi} & & \uparrow{\scriptstyle \phi'} \\ \Sigma I_{\mathbf{COL}} & \xrightarrow[\sigma_{\mathbf{COL}}]{} & \Sigma_{\mathbf{COL}} \end{array}$$

Obviously, any construction that can be used in all general contexts is also usable in all vertical contexts, because any vertical signature morphism is also a general signature morphism. A consequence of that is that the constraint "has to work for all vertical fitting morphisms" is weaker than the constraint "has to work for all fitting morphisms". In conclusion, we should expect that definability and "theorems for free" inferred for the vertical case will be weaker than in the general case. However, one can see the other side of the coin as giving more freedom to the implementer, as he has the possibility of creating more constructions using the knowledge that usage contexts will be restricted.

Restricting usage to vertical contexts improves significantly some of the technical results as we will see in Section 4.3. In particular we will show that globality is synonymous to preservation of *all* correspondences, not just closed ones.

Before advancing any further let us say that all concepts defined for the general case will be redefined in order to reflect the restriction of fitting morphisms to vertical signature morphisms. Hence, we will talk about things like vertical global constructions, vertical correspondences and vertical stability. However, some of the time we will omit the word "vertical" from their name and use it only whenever there is danger of confusion or when we make comparisons to the general case.

4.2 Vertical global constructions

In accordance with the choice of usage contexts, we shall define the appropriate notion of construction that can be used in any vertical context.

Definition 4.1 (Vertical global **COL**-construction)**.** *Let* $k : SPI_{\mathbf{COL}} \Rightarrow SP_{\mathbf{COL}}$ *be a persistent* **COL***-construction. We say that* k *is a* vertical global **COL**-construction *if for every vertical lifting pushout,* $\phi(k)$ *is a* **COL***-construction between* $\phi(SPI_{\mathbf{COL}})$ *and* $\phi'(SP_{\mathbf{COL}})$.

The notion of vertical global construction is closer to what has been studied in the literature in [STar, BST08]. To see that please recall our discussion from Section 2.4 on how the established framework of **OBS** maps into **COL**. In particular, recall that the reachability and observability aspects in the classical frameworks correspond to what we call an integral **COL**-signature, in which every operation acts both as a constructor and as an observer (see Definition 2.34). Recall also that in an integral signature visible sorts are the same as the loose ones and hence the condition of verticality for

OBS-signature morphisms amounts to requiring no change in the nature of the visible sorts, i.e. $\phi(S_{Obs}) \subseteq S'_{Obs}$ (see Section 2.4). The lifting of local constructions into global contexts has been studied in [BST08] relative to **OBS** by using lifting signature morphisms that preserve visible sorts, i.e. vertical according to our definitions. Here we extend that work by considering the full form of **COL**-signatures, not just those that are present in **OBS**, and consequently adopting more general fitting morphisms that take into account also the loose sorts.

Before going any further we will present the typical scenario that involves the reusability of vertical global constructions similarily as it is done for the general case in Proposition 3.4, relative to a vertical lifting pushout.

$$\begin{array}{ccc} SPI'_{\mathbf{COL}} & \xRightarrow{k'} & SP'_{\mathbf{COL}} \\ \Big\Downarrow{\scriptstyle \lceil_{\phi}} & & \Big\Downarrow{\scriptstyle \lceil_{\phi'}} \\ SPI_{\mathbf{COL}} & \xRightarrow[k]{} & SP_{\mathbf{COL}} \end{array}$$

Proposition 4.2 (Lifting vertical constructions)**.** *Let $k : SPI_{\mathbf{COL}} \Rightarrow SP_{\mathbf{COL}}$ be a vertical global* **COL***-construction, and two specifications $SPI'_{\mathbf{COL}}$ and $SP'_{\mathbf{COL}}$ forming a global context connected by the following vertical lifting pushout*

$$\begin{array}{ccc} \mathsf{Sig}[SPI'_{\mathbf{COL}}] & \xrightarrow{\sigma'_{\mathbf{COL}}} & \mathsf{Sig}[SP'_{\mathbf{COL}}] \\ \Big\uparrow{\scriptstyle \phi} & & \Big\uparrow{\scriptstyle \phi'} \\ \mathsf{Sig}[SPI_{\mathbf{COL}}] & \xrightarrow[\sigma_{\mathbf{COL}}]{} & \mathsf{Sig}[SP_{\mathbf{COL}}] \end{array}$$

If the following conditions are satisfied:

1. $SPI'_{\mathbf{COL}} \subseteq \phi(SPI_{\mathbf{COL}})$
2. $\phi(k)(SPI'_{\mathbf{COL}}) \subseteq SP'_{\mathbf{COL}}$

then $k' : SPI'_{\mathbf{COL}} \Rightarrow SP'_{\mathbf{COL}}$, defined as $k'(A') = \phi(k)(A')$ for each $A' \in \mathsf{Mod}[SPI'_{\mathbf{COL}}]$, is a **COL***-construction. Moreover, the following conditions represent an equivalent version of the ones above, but they are more appropriate for practical use because they require the inclusion proofs to be done for a smaller class of algebras.*

1. $RFA(SPI'_{\mathbf{COL}}) \subseteq \phi(SPI_{\mathbf{COL}})$
2. $\phi(k)(RFA(SPI'_{\mathbf{COL}})) \subseteq SP'_{\mathbf{COL}}$

Proof. The fact that k' is a **COL**-construction is immediate from the fact that it is a restriction of the canonical lifting $\phi(k)$.

The interesting part are the sufficient conditions that allow us to consider only the reachable and fully-abstract algebras of the input specification from the global context. The simplification of the first condition is due to the fact that ϕ is a vertical signature morphism, and hence it preserves **COL**-isomorphisms. The simplification of the second condition is due to the fact that $\phi(k)$ is preserving **COL**-isomorphisms, by definition of **COL**-constructions. □

4.2.1 Vertical correspondences

Similarly to the case dealing with general contexts, the globality of the constructions relative to vertical contexts can be captured locally by means of preservation of correspondences. However, the notion of correspondence changes because of the particular properties of the fitting morphisms. General correspondences are just plain algebraic relations, i.e. they are required to be preserved by all of the operations in the signature. Correspondences in the vertical case will carry supplementary information about the loose and visible sorts, as they must capture the essence of observational equalities that come from a vertical extension.

Definition 4.3 (Vertical correspondence)**.** *Let $\Sigma_{\mathbf{COL}} = (S, OP, OP_{Cons}, OP_{Obs})$ be a* **COL***-signature and A, B be two $\Sigma_{\mathbf{COL}}$-algebras. A* (vertical) correspondence ρ *between A and B, written $\rho\colon A \leftrightarrow B$, is an algebraic relation between A and B that is:*

- *bi-injective on visible sorts*
- *bi-surjective on loose sorts.*

If in addition ρ is closed *we call it a (vertical) closed correspondence. Any two elements $a \in A$ and $b \in B$ such that $a\ \rho\ b$ are called (vertical) correspondents.*

We will show that vertical correspondences commute with the observational equalities of the related algebras. This will be the first example where both requirements, bi-surjectivity and bi-injectivity, are used.

Proposition 4.4 (Vertical correspondences commute with observational equalities)**.** *Let $\Sigma_{\mathbf{COL}}$ be a* **COL***-signature, A, B be two $\Sigma_{\mathbf{COL}}$-algebras and $\rho\colon A \leftrightarrow B$ be a vertical correspondence between them. Then for every $a_0, a_1 \in A$ and $b_0, b_1 \in B$ such that $a_0\ \rho\ b_0$, $a_1\ \rho\ b_1$ and $a_0 \approx_{\Sigma_{\mathbf{COL}},A} a_1$ we have that $b_0 \approx_{\Sigma_{\mathbf{COL}},B} b_1$.*

Proof. To show that $b_0 \approx_{\Sigma_{\mathbf{COL}},B} b_1$ we will pick any context $ctx \in \mathcal{C}(\Sigma_{\mathbf{COL}})$ and valuation $\beta : X_{Loose} \to \langle Gen_{\Sigma_{\mathbf{COL}}}(B)\rangle$ and we will prove that $B_{ctx}[\beta](b_0) = B_{ctx}[\beta](b_1)$. For that we consider a corresponding valuation $\alpha : X_{Loose} \to \langle Gen_{\Sigma_{\mathbf{COL}}}(A)\rangle$, i.e. $\alpha(x)\ \rho\ \beta(x)$ for each $x \in X_{Loose}$. Such a valuation exists because ρ is bi-surjective on loose sorts. We now apply the context ctx to the valuation α and to a_0 and a_1 and we obtain that $A_{ctx}[\alpha](a_0) = A_{ctx}[\alpha](a_1)$. Moreover, because α and β correspond through ρ, we get that $A_{ctx}[\alpha](a_0)\ \rho\ B_{ctx}[\beta](b_0)$ and $A_{ctx}[\alpha](a_1)\ \rho\ B_{ctx}[\beta](b_1)$. Finally, because ρ is bi-injective on visible sorts and because we have $A_{ctx}[\alpha](a_0) = A_{ctx}[\alpha](a_1)$, $A_{ctx}[\alpha](a_0)\ \rho\ B_{ctx}[\beta](b_0)$ and $A_{ctx}[\alpha](a_1)\ \rho\ B_{ctx}[\beta](b_1)$ we get that $B_{ctx}[\beta](b_0) = B_{ctx}[\beta](b_1)$. □

The notion of stability, i.e. preservation of correspondences, changes accordingly to the newly introduced notion of correspondence.

Definition 4.5 (Vertically stable **COL**-construction)**.** *Let $k : SPI_{\mathbf{COL}} \Rightarrow SP_{\mathbf{COL}}$ be a tight* **COL***-construction. We say that k is* vertically stable *if it extends vertical correspondences, i.e. for any $SPI_{\mathbf{COL}}$-algebras AI, BI and any vertical correspondence ρ: $AI \leftrightarrow BI$ we have that ρ is a vertical correspondence between $k(AI)$ and $k(BI)$. We say that k is* vertically closed-stable *if it extends closed vertical correspondences.*

Similarly to the general case we can prove that closed-stability implies globality, i.e. preservation of closed vertical correspondences is sufficient in order to lift a local construction to any vertically fitted context.

Proposition 4.6 (Closed-stability implies globality)**.** *Let $k : SPI_{\mathbf{COL}} \Rightarrow SP_{\mathbf{COL}}$ be a tight* **COL***-construction. If k is vertically closed-stable then k is a vertical global* **COL***-construction.*

Proof. Consider a vertical lifting pushout as below:

$$\begin{array}{ccc} \Sigma I'_{\mathbf{COL}} & \xrightarrow{\sigma'_{\mathbf{COL}}} & \Sigma'_{\mathbf{COL}} \\ \phi \uparrow & & \uparrow \phi' \\ \Sigma I_{\mathbf{COL}} & \xrightarrow[\sigma_{\mathbf{COL}}]{} & \Sigma_{\mathbf{COL}} \end{array}$$

We need to show that $\phi(k)$ is a **COL**-construction along $\sigma'_{\mathbf{COL}}$ between $\phi(SPI_{\mathbf{COL}})$ and $\phi'(SP_{\mathbf{COL}})$. For simplicity we will write k' instead of $\phi(k)$.

First we will show that $k'(AI') \in \mathsf{Alg}_{\mathbf{COL}}(\Sigma'_{\mathbf{COL}})$ for all $AI' \in \phi(SPI_{\mathbf{COL}})$. Let $AI = AI'\restriction_{\phi}$, $A = k(AI)$ and $A' = k'(AI')$. We claim that the constraints imposed by $\Sigma I'_{\mathbf{COL}}$ and $\Sigma'_{\mathbf{COL}}$, on AI' and respectively A', coincide, i.e. $\langle Gen_{\Sigma I'_{\mathbf{COL}}}(AI')\rangle = \langle Gen_{\Sigma'_{\mathbf{COL}}}(A')\rangle\restriction_{\sigma'}$ and $\approx_{\Sigma I'_{\mathbf{COL}},AI'} = \approx_{\Sigma'_{\mathbf{COL}},A'}\restriction_{\sigma'}$.

Notice that because $\sigma'_{\mathbf{COL}}$ is a **COL** signature morphism, i.e. it does not add new constructors or observers, we get that $Gen_{\Sigma I'_{\mathbf{COL}}}(AI') = Gen_{\Sigma'_{\mathbf{COL}}}(A')\restriction_{\sigma'}$ and that $\approx_{\Sigma I'_{\mathbf{COL}},AI'} = \approx_{\Sigma'_{\mathbf{COL}},A'}\restriction_{\sigma'}$. We will show that $\langle Gen_{\Sigma I'_{\mathbf{COL}}}(AI')\rangle = \langle Gen_{\Sigma'_{\mathbf{COL}}}(A')\rangle\restriction_{\sigma'}$ and that $\approx_{\Sigma'_{\mathbf{COL}},A'}$ is a congruence on $\langle Gen_{\Sigma'_{\mathbf{COL}}}(A')\rangle$. For that let $\rho\colon AI \leftrightarrow AI$ be the restriction of $\approx_{\Sigma I'_{\mathbf{COL}},AI'}$ to $\langle Gen_{\Sigma I'_{\mathbf{COL}}}(AI')\rangle$ and reduced via ϕ. It is easy to see that ρ is a closed vertical correspondence and because k is vertically closed-stable we get that ρ is preserved by k and hence the operations added by $\sigma_{\mathbf{COL}}$ commute with ρ. That means in particular that the elements produced by those operations are still in $\langle Gen_{\Sigma I'_{\mathbf{COL}}}(AI')\rangle$ and hence $\langle Gen_{\Sigma I'_{\mathbf{COL}}}(AI')\rangle = \langle Gen_{\Sigma'_{\mathbf{COL}}}(A')\rangle\restriction_{\sigma'}$. Now, we can see that ρ is a congruence on the generated algebra $\langle Gen_{\Sigma'_{\mathbf{COL}}}(A')\rangle$ by noticing that the operations that come from $\Sigma'_{\mathbf{COL}}$ preserve $\approx$ by default and the ones added by σ preserve it due to stability.

Secondly we need to show that for all $AI' \equiv_{\Sigma I'_{\mathbf{COL}}} BI'$ we have $k(AI') \equiv_{\Sigma'_{\mathbf{COL}}} k(BI')$. For that let h be the isomorphism between AI' and BI' and ρ be the correspondence equal to $h\restriction_{\phi}$. It is trivial that ρ is a closed vertical correspondence, and using the fact that k is vertically closed-stable we get that ρ commutes with the added operations. Finally, because σ is tight we can conclude that h is a **COL**-isomorphism between $k'(AI')$ and $k'(BI')$. □

Now, we will revise the proof of Proposition 3.8 in order to obtain the other direction of the implication, i.e. that closed vertical correspondences are necessarily extended by vertical global constructions.

Proposition 4.7 (Globality implies closed-stability)**.** *Let $k : SPI_{\mathbf{COL}} \Rightarrow SP_{\mathbf{COL}}$ be a tight* **COL***-construction. If k is a vertical global construction then k is vertically closed-stable.*

Proof. Let AI and BI be two $SPI_{\mathbf{COL}}$ models and let ρ be a closed vertical correspondence between AI and BI. Similarly to what we have done for the general contexts' case we will build a global context $\Sigma I'_{\mathbf{COL}}$ and two $\Sigma I'_{\mathbf{COL}}$-isomorphic extensions AI' and BI' of AI, and respectively BI, such that the corresponding isomorphism extends ρ. We will stress only those details that differ from the proof of Proposition 3.8.

We define $\Sigma I'_{\mathbf{COL}}$ to be equal to $(SI', OPI', OPI'_{Cons}, OPI'_{Obs})$ where

- $SI' = SI \uplus \{Bool\}$
- $OPI' = OPI \uplus \{!^{a,b} : s \mid (a,b) \in \rho_s, s \in SI\} \uplus \{?^b : s \to Bool \mid b \in B_s, s \in SI_{State}\} \uplus \{true, false : Bool\}$

- $OPI'_{Cons} = \{!^{a,b} : s \mid (a,b) \in \rho_s, s \in SI_{Cons}\} \uplus \{true, false : Bool\}$
- $OPI'_{Obs} = \{(op,i) \mid op : s_1,\ldots,s_n \to s \in OPI, s_i \in SI_{State}, 1 \leq i \leq n\} \uplus \{?^b : s \to Bool \mid b \in B_s, s \in SI_{State}\}$

Please notice that we have defined the observers and constructors for $\Sigma I'$ in such a way that the fitting morphism ϕ, i.e. the inclusion from ΣI to $\Sigma I'$, is a vertical morphism, i.e. visible and loose sorts do not change their nature. From that definition we can see that $SI'_{Loose} = SI_{Loose}$, $SI_{Cons} = SI_{Cons} \uplus \{Bool\}$, $SI'_{State} = SI_{State}$ and $SI'_{Obs} = SI_{Obs} \uplus \{Bool\}$. Notice the differences between this definition and that present in the proof of Proposition 3.8 where all sorts in $\Sigma'_{\mathbf{COL}}$ were defined to be constructed and also hidden (with the exception of *Bool*).

Now, let us define AI' such that $AI'|_\phi = AI$ and BI' such that $BI'|_\phi = BI$. For that we must define only the interpretation of the additional symbols introduced by ϕ. We let the sort *Bool* be interpreted canonically, i.e. $AI'_{Bool} = BI'_{Bool} = \{true, false\}$, and for the additional operations we use the same interpretation as in the general case:

AI'**:**
- $!^{a,b} = a$ for all $(a,b) \in \rho$
- $?^b(a) = true$ if $a\ \rho\ b$ and *false* otherwise

BI'**:**
- $!^{a,b} = b$ for all $(a,b) \in \rho$
- $?^b(b') = true$ if there exists a such that $a\ \rho\ b$ and $a\ \rho\ b'$ and *false* otherwise

Following the same reasoning as in the proof of Proposition 3.8 we can show that the only relation between AI' and BI' that extends ρ and is identity on *Bool* is a **COL**-isomorphism. Furthermore, we know that $\phi(k)(AI') \equiv_{\Sigma'_{\mathbf{COL}}} \phi(k)(BI')$ and hence the corresponding isomorphism is an extension of ρ. Finally, that means that ρ is an algebraic relation between $k(AI)$ and $k(BI)$. □

Please notice that in the general case (see Proposition 3.8), all algebraic relations needed to be preserved in order to ensure globality. In the vertical case, due to the smaller class of contexts in which a construction might be used, we obtain that a smaller class of relations are necessarily preserved. More explicitly, vertical global constructions need not preserve algebraic relations that do not map all elements of loose sorts because they will not be used in contexts in which the generated parts on those sorts will be a strict subset of the corresponding carriers. Also, the relations that are not one to one correspondences on visible sorts cannot appear as **COL**-isomorphisms between algebras in contexts that preserve the nature of those visible sorts.

We can now put together the two propositions concerning necessity and sufficiency of closed-stability for globality, in order to obtain a characterization of global constructions by means of preservation of closed-correspondences.

Theorem 4.8 (Globality is equivalent to closed-stability)**.** *Let $k : SPI_{\mathbf{COL}} \Rightarrow SP_{\mathbf{COL}}$ be a tight* **COL***-construction. Then k is vertically global if and only it is vertically closed-stable.*

Proof. A direct consequence of Proposition 4.6 and Proposition 4.7. □

4.2.2 Definability

Because a smaller number of usage contexts means more flexibility granted to the implementer, the definability results for the vertical case will not be as strong as in the general case. We anticipate the results by saying that we will obtain a definability result for vertical **COL**-constructions, but the defining terms will contain not only operations from the source signature but also elements from the carriers of the loose sorts induced by that signature. To express this formally we have to revise our definition of definability in order to make it aware of the additional information present in a **COL**-signature.

Definition 4.9 (Loose definability)**.** *Let $\Sigma_{\mathbf{COL}}$ and $\Sigma'_{\mathbf{COL}}$ be two* **COL***-signatures and let $\sigma_{\mathbf{COL}}$ be a tight* **COL***-signature morphism between them, i.e. $\sigma_{\mathbf{COL}} : \Sigma_{\mathbf{COL}} \to \Sigma'_{\mathbf{COL}}$. Let $op' : s_1, \ldots, s_n \to s \in OP'$ be an operation symbol and A' be a $\Sigma'_{\mathbf{COL}}$-algebra.*

We say that op' is (point-wise) loose definable *on A' if for every tuple $a_1, \ldots, a_n$, with $a_i \in A'_{s_i}$ for $i = 1 \ldots n$, there exists a term $t \in T_{\Sigma \cup A'_{Loose}}(\{x_1, \ldots, x_n\})$ such that:*

$$A'_{op'}(a_1, \ldots, a_n) = A'_t[a_1/x_1, \ldots, a_n/x_n]$$

We will call the term t the defining term *for op' and $a_1, \ldots, a_n$. (The definition extends easily to sets of operations and classes of algebras).*

Let $k : SPI_{\mathbf{COL}} \Rightarrow SP_{\mathbf{COL}}$ be a tight **COL***-construction. We say that k is* loose definable *if all operations from OP are loose definable on $k(SPI_{\mathbf{COL}})$.*

In this section we will use mainly the notion of loose definability and for simplicity we will just call it definability when there is no danger of confusion with the notion of algebraic definability used in Chapter 3.

Now, we can use this concept in order to show that all vertically closed-stable constructions are definable.

Proposition 4.10 (Closed-stability implies loose definability)**.** *Let $k : SPI_{\mathbf{COL}} \Rightarrow SP_{\mathbf{COL}}$ be a tight* **COL***-construction. If k is vertically closed-stable then k is loose definable.*

Proof. Let A be a **COL**-algebra in $k(SPI_{\mathbf{COL}})$, i.e. there exists $AI \in SPI_{\mathbf{COL}}$ such that $k(AI) = A$. Let $op : w \to s \in OP$ and consider a tuple $\underline{a} \in A_w$. We define the vertical correspondence $\rho : AI \leftrightarrow AI$ that relates $AI_t[\underline{a}]\ \rho\ AI_t[\underline{a}]$ for all $t \in T_{\Sigma \cup AI_{Loose}}(\{\underline{x}\})$. Note that ρ is a closed vertical correspondence because all elements from AI_{Loose} are related and also the fact that $\rho \subseteq =$ makes it closed and injective on visible sorts. Because k extends closed vertical correspondences we get that $A_{op}(\underline{a})\ \rho\ A_{op}(\underline{a})$ and therefore, by the way we defined ρ, there exists $t \in T_{\Sigma \cup AI_{Loose}}(\{\underline{x}\})$ such that $A_{op}(\underline{a}) = A_t[\underline{a}]$. □

Similarly to the general case we can produce a sound criteria for proving that a construction is closed-stable. We have proved in Proposition 3.13 that uniform general definable constructions, i.e. those for which the implemented operations are definable by the same defining term when applied to generally correspondent arguments, are generally closed-stable. In the vertical case we can prove a similar result by requiring that the defining terms are correspondent, not necessarily equal, for vertically correspondent arguments. In order to express that we will first define what we mean by correspondent terms.

Definition 4.11 (Correspondent terms)**.** *Let $\Sigma_{\mathbf{COL}}$ be a* **COL***-signature, A and B be two $\Sigma_{\mathbf{COL}}$-algebras and let ρ be a vertical correspondence between A and B. We say that two terms $t_0 \in T_{\Sigma \cup A_{Loose}}(X)$ and $t_1 \in T_{\Sigma \cup B_{Loose}}(X)$ are correspondent via ρ if there exists a set of variables Y, a term $t \in T_{\Sigma \cup Y}(X)$ and two ρ-correspondent valuations $\alpha : Y \to A_{Loose}$ and $\beta : Y \to B_{Loose}$, i.e. $\alpha(y)\ \rho\ \beta(y)$ for all $y \in Y$, such that $t[\alpha] = t_0$ and $t[\beta] = t_1$.*

Basically, two terms with variables from X and elements from the loose carriers of A and B are correspondent if they have a common "shape" (represented in the definition by the term t) and the loose elements that are present in their body correspond one to another.

We can now express the notion of uniform definability for vertical correspondents.

Definition 4.12 (Uniform loose definability)**.** *Let $\Sigma_{\mathbf{COL}}$ and $\Sigma'_{\mathbf{COL}}$ be two* **COL***-signatures and let $\sigma_{\mathbf{COL}}$ be a tight* **COL***-signature morphism $\sigma_{\mathbf{COL}} : \Sigma_{\mathbf{COL}} \to \Sigma'_{\mathbf{COL}}$. We say that an operation $op' : w \to s \in OP'$ is* uniform loose definable *on a class $\mathcal{A}'$ of $\Sigma'_{\mathbf{COL}}$-algebras if for all $A', B' \in \mathcal{A}'$ and all correspondent tuples $\underline{a} \in A'_w$ and $\underline{b} \in B'_w$, via*

a closed vertical $\Sigma_{\mathbf{COL}}$*-correspondence* ρ*, there exists two* ρ*-correspondent defining terms,* t_0 *for* op' *and* $\underline{a}$*, and* t_1 *for* op' *and* $\underline{b}$*.*

Uniform definability in the vertical case is conceptually similar to uniform definability in the general case. However, we should have in mind that they are based on different kinds of correspondences, and while there is a general correspondence between any two elements of every two algebras that is not the case with vertical correspondences. For example, the vertical correspondence's requirement of being bi-injective on the visible sorts typically stops one to put in correspondence elements that have different observable behaviors.

Proposition 4.13 (Uniform definability implies closed-stability)**.** *Let* $k : SPI_{\mathbf{COL}} \Rightarrow SP_{\mathbf{COL}}$ *be a tight* **COL***-construction. If k is uniform loose definable then k is vertically closed-stable.*

Proof. Let AI, BI be two $\Sigma I_{\mathbf{COL}}$-algebras and ρ: $AI \leftrightarrow BI$ be a closed vertical correspondence between them. We will denote by A and B the constructed algebras $k(AI)$ and $k(BI)$. Also, consider an operation $op : w \to s \in OP$ and two ρ-correspondent tuples $\underline{a} \in A_w$ and $\underline{b} \in B_w$. Using the uniform loose definability assumption we get that there are two correspondent terms, t_0 for op and $\underline{a}$ and t_1 for op and $\underline{b}$, i.e. $A_{op}(\underline{a}) = A_{t_0}[\underline{a}]$ and $B_{op}(\underline{b}) = B_{t_1}[\underline{b}]$. Let t be the common "shape" of the two correspondent defining terms t_0 and t_1 with α and β being the two correspondent valuations such that $t_0 = t[\alpha]$ and $t_1 = t[\beta]$. Because ρ is an algebraic relation, i.e. it is preserved by the operations in the ΣI signature, we get that $A_t[\alpha][\underline{a}] \; \rho \; B_t[\beta][\underline{b}]$. That is equivalent to $A_{t_0}[\underline{a}] \; \rho \; B_{t_1}[\underline{b}]$ and hence the interpretations of op in A and B preserve the relation ρ. As that happens for every operation and related arguments we get that ρ is a Σ-correspondence and hence k extends ρ. □

Is the reverse implication true, i.e. is every closed-stable construction uniformly definable? We can produce a counterexample for that statement. Hence, it is not the case that any vertically closed-stable construction is uniform loose definable. The problem is that term generated relations, which could have been used to prove the uniformity, are not typically closed and therefore are not necessarily extended by closed-stable constructions. However, we will see in Section 4.3 that under the condition that the source specification is iso-closed we can prove that closed stability is equivalent to stability and does imply uniform definability. That means that we cannot provide the same counterexample as in Example 3.15 for the failure of uniform definability (be-

cause that construction was defined for an iso-closed class of **COL**-algebras) and we must find a construction that acts on a class that is not iso-closed.

Example 4.14 (Closed stability does not imply uniform definability)**.** *We will define a tight construction $k : SPI_{\mathbf{COL}} \Rightarrow SP_{\mathbf{COL}}$ that is vertically closed stable but is not uniform loose definable. For that let $\Sigma I_{\mathbf{COL}}$ be the following signature:*

sorts s, v

operations $out : s \to v$

$x_{00}, x_{01}, x_{10} : s$

$\bot : v$

constructors $\bot, x_{00}, x_{01}, x_{10}$

observers out

and let $\Sigma_{\mathbf{COL}}$ be the signature that adds a constant $x_{11} : s$ to $\Sigma I_{\mathbf{COL}}$.

Now let $\mathsf{Mod}[SPI_{\mathbf{COL}}] = \{A, B\}$, i.e. the semantics of the specification is given just by the two **COL***-algebras A defined as follows:*

$$A_s = B_s = \{c_0, c_1\} \text{ and } A_v = B_v = \{\bot\}$$

$$A_{x_{00}} = c_0,\ A_{x_{01}} = c_0,\ A_{x_{10}} = c_1$$

$$B_{x_{00}} = c_0,\ B_{x_{01}} = c_1,\ B_{x_{10}} = c_0$$

A construction $k : SPI_{\mathbf{COL}} \Rightarrow SP_{\mathbf{COL}}$ must implement $x_{11} : s$ for each algebra $AI \in SPI_{\mathbf{COL}}$. Because we only have two algebras in $SPI_{\mathbf{COL}}$ we can define it directly by letting $A_{x_{11}} = B_{x_{11}} = c_1$.

The defined construction is closed-stable because any closed algebraic relation ρ between A and B must obey $c_0\ \rho\ c_0$, $c_0\ \rho\ c_1$, $c_1\ \rho\ c_0$ by algebraicity and then imediately $c_1\ \rho\ c_1$ by closedness. Therefore k extends all closed algebraic relations, i.e. it is generally closed-stable and in particular it is vertically closed stable. However, k is not uniformly definable because there is no term that evaluates to c_1 in both algebras.

Please notice that the counterexample we are presenting does not use the particular requirements of the vertical case as we have no loose sorts. In fact this is a perfectly good counterexample for the general case as well. The reason we preferred to present a more elaborate counterexample in Example 3.15 was to show that, in the general case, we can find counterexamples even if the source specification is iso-closed. That is not possible in the vertical case as we will see in Section 4.3.

4.2.3 Theorems for free

We will now look at some concrete examples of how vertical closed stability can be used to derive useful properties about vertical global constructions. In the general case (see Section 3.5) we worked with a specification of lists as inhabitants of a loose and visible sort in the local context, because we had the liberty to change the nature of that sort in global contexts. In the vertical case, that is not possible anymore and we will change the setting by considering a specification of containers as inhabitants of a hidden and constructed sort. Our example will contain specifications CONTAINERWITHNAME and CONTAINERWITHNAMEANDREPLACE for the global context:

spec CONTAINERWITHNAME = BOOL **THEN**
  sorts $Container, Elem, String$
  operations $empty : Container$
    $insert, remove : Elem, Container \to Container$
    $isIn : Elem, Container \to Bool$
    $name : Container \to String$
    $change : String, Container \to Container$
  constructors $empty, insert, change$
  observers $(isIn, 2), (name)$
  axioms $name(change(s,c)) = s$
    $name(insert(x,c)) = name(c)$
    $name(remove(x,c)) = name(c)$
    $isIn(x, insert(y,c)) = x = y \vee isIn(x,c)$
    $isIn(x, remove(y,c)) = x \neq y \wedge isIn(x,c)$
end

spec CONTAINERWITHNAMEANDREPLACE = CONTAINERWITHNAME **THEN**
  operations $replace : Elem, Elem, Container \to Container$
  axioms **% This is a theorem for free**
    $name(replace(x,y,c)) = name(c) \vee name(replace(x,y,c)) = name(empty)$
end

and specifications CONTAINERSKELETON and CONTAINERSKELETONWITHREPLACE

for the local context:

spec CONTAINERSKELETON = BOOL **THEN**
  sorts *Container*, *Elem*
  operations *empty* : *Container*
    insert, *remove* : *Elem*, *Container* → *Container*
    isIn : *Elem*, *Container* → *Bool*
  constructors *empty*, *insert*
  observers (*isIn*, 2)
end

spec CONTAINERSKELETONWITHREPLACE = CONTAINERSKELETON **THEN**
  operations *replace* : *Elem*, *Elem*, *Container* → *Container*
end

whose signatures are related by the following pushout:

$$\begin{array}{ccc} \mathsf{Sig}[\text{CONTAINERWITHNAME}] & \longrightarrow & \mathsf{Sig}[\text{CONTAINERWITHNAMEANDREPLACE}] \\ \uparrow \phi & & \uparrow \phi' \\ \mathsf{Sig}[\text{CONTAINERSKELETON}] & \longrightarrow & \mathsf{Sig}[\text{CONTAINERSKELETONWITHREPLACE}] \end{array}$$

The local context based on CONTAINERSKELETON presents containers generated by *empty* and *insert* and observed by *isIn*, hence modeling finite sets of elements. The global context introduces in addition a new observation (*name*) that returns a name for a container and a new constructor (*change*) that changes the name of a container. However, the nature of loose or visible sorts (in our case *Elem*) is not changed and hence the fitting morphism is vertical.

According to Proposition 4.2, in order to lift a vertical global construction k : CONTAINERSKELETON⇒CONTAINERSKELETONWITHREPLACE to the global context it is sufficient to prove:

- $RFA(\text{CONTAINERWITHNAME}) \subseteq \phi(\text{CONTAINERSKELETON})$
- $\phi(k)(RFA(\text{CONTAINERWITHNAME})) \subseteq SP'_{\mathbf{COL}}$

In order to prove the first inclusion we have to show that reducts of reachable and fully abstract models in CONTAINERWITHNAME satisfy the **COL**-constraints imposed by the local context. We will not give that proof as it is not the point of our

example, but we will remark that the axioms defining the result of observation *isIn* after a *remove* or an *insert* operation allow us to prove that reachability and observability constraints are satisfied by those operations.

For the second inclusion, similarly to the example in Section 3.5, we have that *insert* and *remove* preserve the names of the containers. Definability in this case means that the implemented operation *replace* should be definable in terms of *insert*, *remove* and *empty*. Because *insert* and *remove* preserve names we can easily conclude that *replace* either produces a container with the name of the *empty* container or it preserves the name of its argument.

4.3 Vertical global constructions on iso-closed classes

Typical specifications are at least iso-closed, in particular basic specifications induced by a set of axioms are iso-closed. We can draw some useful consequences about the nature of the defining terms for constructions that start from a iso-closed specification. We have seen already that in the vertical case, loose definability ensures us that for any implemented operation we can find a defining term that contains elements from the loose carriers of the algebras. While that result can still be exploited to gain more information about the implemented operations, it is useless for reasoning about the operations that have a loose result sort. However, using additional information, like the requirement for closedness under behavioral isomorphisms, we can strengthen significantly the result that can be obtained. In that case, we will always be able to find a defining term that contains elements only from the carriers of visible sorts.

First we will show that any (plain) vertical correspondence can be decomposed into two closed vertical correspondences using an intermediate algebra that is **COL**-isomorphic to the original algebras.

Proposition 4.15 (Decomposition of vertical correspondences)**.** *Let* $\Sigma_{\mathbf{COL}}$ *be a* **COL***-signature,* A, B *be two* $\Sigma_{\mathbf{COL}}$*-algebras and* ρ *be a vertical correspondence between them. Then there exists a* $\Sigma_{\mathbf{COL}}$*-algebra* T_ρ *such that* $A \equiv_{\Sigma_{\mathbf{COL}}} T_\rho \equiv_{\Sigma_{\mathbf{COL}}} B$*, and two vertical closed correspondences* $\rho_A \colon A \leftrightarrow T_\rho$ *and* $\rho_B \colon T_\rho \leftrightarrow B$ *such that* $\rho_A ; \rho_B = \rho$.

Proof. Let T_ρ contain all the pairs that are linked by the relation ρ, i.e. $(T_\rho)_s = \{(a,b) \mid (a,b) \in \rho_s\}$ for all $s \in S$. The operations are defined component-wise and we can see that A is well-defined as a Σ-algebra because ρ is algebraic. It is easy to check that observational equality on T_ρ decomposes on two components, i.e. $(a_0, b_0) \approx_{\Sigma_{\mathbf{COL}}, T_\rho}$

(a_1, b_1) if and only if $a_0 \approx_{\Sigma_{\mathbf{COL}},A} a_1$ and $b_0 \approx_{\Sigma_{\mathbf{COL}},B} b_1$. However using Proposition 4.4 it is sufficient for the observational equality to hold just for one of these to enforce the equality of the pair, i.e. $a_0 \approx_{\Sigma_{\mathbf{COL}},A} a_1$ implies $b_0 \approx_{\Sigma_{\mathbf{COL}},B} b_1$ and vice versa. Using that we can easily show that T_ρ obeys the reachability constraint because for each element in the generated algebra $(a_0, b_0) \in \langle Gen_{\Sigma_{\mathbf{COL}}}(T_\rho)\rangle$ we can find an observationally equal element in the generated part $(a_1, b_1) \in Gen_{\Sigma_{\mathbf{COL}}}(T_\rho)$ just by focusing on finding the constructor term that evaluates to something observationally equal to a_0. The observability constraint trivially follows from the component-wise preservation of the observational equality.

To see that $A \equiv_{\Sigma_{\mathbf{COL}}} T_\rho$ we define the **COL**-morphism $h\colon A \to T_\rho$ as $a_0\ h\ (a_1, b_1)$ if and only if $a_0 \approx_{\Sigma_{\mathbf{COL}},A} a_1$ for elements in the generated algebras. We can easily check that all conditions in the definition of the **COL**-morphism are satisfied.

In order to define the two closed vertical correspondences ρ_A and ρ_B, such that $\rho{=}\rho_A;\rho_B$, just take $a\ \rho_A\ (a,b)$ and $(a,b)\ \rho_B\ b$ for all $(a,b) \in \rho$. □

We can now prove that globality does not just imply preservation of closed vertical correspondences but, more generally, it implies preservation of *all* vertical correspondences.

Proposition 4.16 (Globality implies stability)**.** *Let $k : SPI_{\mathbf{COL}} \Rightarrow SP_{\mathbf{COL}}$ be a tight* **COL***-construction such that $SPI_{\mathbf{COL}}$ is iso-closed. Then k is vertically stable if it is vertically global.*

Proof. To prove that any vertical correspondence is preserved we first decompose it into two closed vertical correspondences using Proposition 4.15. Then using the closed-stability, guaranteed by Proposition 4.7, we obtain that those are preserved, and therefore their composition will also be preserved. □

We can now state the equivalence between vertical globality and vertical stability.

Theorem 4.17 (Globality is equivalent to stability)**.** *Let $k : SPI_{\mathbf{COL}} \Rightarrow SP_{\mathbf{COL}}$ be a tight* **COL***-construction such that $SPI_{\mathbf{COL}}$ is iso-closed. Then k is vertically global if and only it is vertically stable.*

Proof. One implication has just be proven in Proposition 4.16, and for the other implication we use the fact that stability is stronger than closed-stability which is sufficient to derive globality (see Proposition 4.6). □

Another improvement, due to preservation of all vertical correspondences, is that for iso-closed classes the uniform loose definability of global constructions becomes necessary.

Proposition 4.18 (Globality implies uniform loose definability)**.** *Let* $k : SPI_{\mathbf{COL}} \Rightarrow SP_{\mathbf{COL}}$ *be a tight* **COL***-construction such that* $SPI_{\mathbf{COL}}$ *is iso-closed. Then k is uniform loose definable if it is vertically global.*

Proof. Let AI, BI be two $\Sigma I_{\mathbf{COL}}$-algebras, let $op : w \to s \in OP$ and let $\underline{a} \in AI_w$ and $\underline{b} \in BI_w$ be two correspondent elements via a vertical correspondence ρ: $AI \leftrightarrow BI$. We define another vertical correspondence ρ_0: $AI \leftrightarrow BI$ that relates $AI_{t_0}[\underline{a}]\ \rho_0\ BI_{t_1}[\underline{b}]$ for all ρ-correspondent $t_0 \in T_{\Sigma \cup AI_{Loose}}(\{\underline{x}\})$ and $t_1 \in T_{\Sigma \cup BI_{Loose}}(\{\underline{x}\})$. Because k extends ρ_0 we can conclude that there exists two ρ-correspondent terms t_0, t_1 that define op for $\underline{a}$ and resp. $\underline{b}$. □

With the previous proposition we have proved that vertical globality is characterized, soundly and completely, by uniform loose definability, and we will write these conclusions in the next theorem.

Theorem 4.19 (Globality is equivalent to uniform loose definability)**.** *Let* $k : SPI_{\mathbf{COL}} \Rightarrow SP_{\mathbf{COL}}$ *be a tight* **COL***-construction such that* $SPI_{\mathbf{COL}}$ *is iso-closed. Then k is vertically global if and only it is uniformly loose definable.*

Proof. A direct consequence of Proposition 4.13 and of Proposition 4.18. □

We have proved until now that working with iso-closed classes ensures us that global constructions can be characterized in terms of stability or uniform loose definability. We will now focus on some useful properties that will lead us towards a stronger definability result.

First we define a special kind of algebraic relations called *self-correspondences*, and then we will show that these are preserved by vertical global constructions, even though they are not vertical correspondences. As the name suggests, we will require that a self-correspondence relates an algebra to itself, it only relates equal elements and it must relate all "relevant" visible elements.

Definition 4.20 (Self-correspondence)**.** *Let* $\Sigma_{\mathbf{COL}}$ *be a* **COL***-signature and let A be a* $\Sigma_{\mathbf{COL}}$*-algebra. A* self-correspondence ρ *on A is an algebraic relation between A and A such that:*

- $a\ \rho\ b$ *implies* $a = b$

- $\langle Gen^{dom(\rho)}_{\Sigma_{\mathbf{COL}}}(A)\rangle_v \subseteq dom(\rho_v)$ *for all* $v \in S_{Obs}$

In the previous definition we have used the notion of minimal generated part (see Definition 2.13), that is $\langle Gen^{dom(\rho)}_{\Sigma_{\mathbf{COL}}}(A)\rangle$ is the minimal generated part that contains the elements from $dom(\rho)$.

One can see that self-correspondences are in fact a special kind of predicate. This kind of predicate can be defined algebraically by identifying a class of algebras. If we denote by *loose subalgebras* those subalgebras that preserve the carriers of loose sorts, and by *visible subalgebras* those subalgebras that preserve the carriers of visible sorts, then a self-correspondences for a **COL**-algebra A is a visible subalgebra of a loose subalgebra of A. That means that for a self-correspondence it is not important to contain all loose sorts, but should contain all their visible outcomes.

We will now show that vertical global constructions which start from an iso-closed specification must extend self-correspondences. The proof is done by completing a self-correspondence to a standard correspondence, by relating each loose element to itself. Because not all loose elements are guaranteed to be present in the original self-correspondence, we will have to force them into the complete correspondence. But, we will also want to keep track of those elements that were originally present in the self-correspondence. That is why we use a helper algebra in which the loose elements that were not in the self-correspondence from the beginning will be tagged in order to distinguish them. In fact our complete correspondence will be defined on the helper algebra, and based on the fact that correspondences are extended by generic constructions, and on the fact that the untagged elements form a stable partition of the complete correspondence, we will obtain that the original self-correspondence is also preserved.

Proposition 4.21 (Extending self-correspondences)**.** *Let* $k : SPI_{\mathbf{COL}} \Rightarrow SP_{\mathbf{COL}}$ *be a tight* **COL***-construction such that* $SPI_{\mathbf{COL}}$ *is iso-closed. If* k *is a global vertical construction then* k *extends all self-correspondences.*

Proof. Let $AI \in SPI_{\mathbf{COL}}$ and ρ: $AI \leftrightarrow AI$ be a self-correspondence. We need to show that k extends it, i.e. that ρ is an algebraic relation on $A = k(AI)$.

In order to prove the preservation of a self-correspondence by a global construction we will base our arguments on the preservation of vertical correspondences. Therefore, we will try to complete ρ to a vertical correspondence, i.e. one that is bi-surjective on all loose sorts. However, we cannot do that directly on AI, so we will construct a helper algebra T and do the completion on it.

Before proceeding any further please notice that $\langle Gen_{\Sigma I_{\mathbf{COL}}}^{dom(\rho)}(AI)\rangle$ is a $\Sigma I_{\mathbf{COL}}$-algebra that is isomorphic to AI. As we can also see ρ as a relation on $\langle Gen_{\Sigma I_{\mathbf{COL}}}^{dom(\rho)}(AI)\rangle$, we will assume w.l.o.g. that $AI = \langle Gen_{\Sigma I_{\mathbf{COL}}}^{dom(\rho)}(AI)\rangle$ discarding the elements that are not relevant neither to the algebra's semantics nor to the relation ρ.

The helper algebra T will be defined as follows. On visible sorts it will be exactly like AI, i.e. $T_v = AI_v$ for all $v \in SI_{Obs}$. On a hidden sort s it will contain two tagged copies for each element $a \in AI_s$, and we will write a^+ and a^- to distinguish these copies. It will also contain a copy for each element a in the domain of ρ_s which will be written simply as a. We will call the elements we have described so far, both on visible and on hidden sorts, *direct copies*. In addition to the direct copies, the hidden carriers will contain all the *tagged terms* over direct copies (the tagged terms are Σ-terms over the direct copies that contain at least a tagged element). To sum up, the visible sorts are interpreted as in AI, while the hidden sorts contain multiple direct copies (tagged and untagged) of the elements in AI and terms (only tagged) over these copies. Please note that the only hidden untagged elements in T are those that are present in $dom(\rho)$. That is true also for visible sorts, because from the second requirement of the definition of self-correspondences and from our assumption that $AI = \langle Gen_{\Sigma I_{\mathbf{COL}}}^{dom(\rho)}(AI)\rangle$ we get $AI_v = dom(\rho_v)$ for all $v \in SI_{Obs}$. To conclude, on all sorts the elements in T are untagged if and only if they are $dom(\rho)$. For all elements in T we can define a *standard value*, that belongs to AI, by forgetting the tags and evaluating the obtained term in AI. We write $\hat{t} \in AI$ for the standard value of $t \in T$.

The interpretation of the operations depends firstly on the result sort. If the result sort is visible, then $T_{op}(\underline{t}) = AI_{op}(\underline{\hat{t}})$. If the result sort is hidden we treat two cases. If one of the arguments is tagged, either tagged direct copy or tagged term, then the result will be a tagged term $T_{op}(\underline{t}) = op(\underline{t})$. If none of the arguments is tagged the result will be untagged and obtained by evaluating the operation on the arguments in AI, i.e. $T_{op}(\underline{a}) = AI_{op}(\underline{a})$. The defined algebra is obviously observationally isomorphic with AI, as it contains just additional copies of the already existing behaviors, and hence the vertical correspondences on T are extended by k. Therefore, we can focus our attention on vertical correspondences on T, and we will define one that contains the original self-correspondence ρ.

For that, let $\rho^T \colon T \leftrightarrow T$ be defined as follows. First we want ρ^T to relate all untagged elements to themselves, and hence we will have that $a\ \rho\ a$ implies $a\ \rho^T\ a$ for all $a \in AI$. Secondly, the tagged elements are related by changing the tags, i.e. $a^+\ \rho^T\ a^-$ and $a^-\ \rho^T\ a^+$ for any direct tagged copies. Two tagged terms are related by ρ^T if

and only if they have the same structure but corresponding direct tagged copies in that structure have opposite tags. We can see easily that ρ^T is a vertical correspondence on T. The main property of ρ^T is that it does not relate any tagged element to itself; if one element is related to itself it must be untagged and must be in $dom(\rho)$, i.e. $t\ \rho^T\ t$ implies $t \in dom(\rho)$ and $t\ \rho\ t$ for all $t \in T$.

Now consider an operation symbol $op : w \to s$ added by the tight signature morphism $\Sigma I_{\mathbf{COL}} \to \Sigma_{\mathbf{COL}}$ and let $\underline{a}, \underline{b} \in AI$ be such that $\underline{a}\ \rho\ \underline{b}$. However because ρ is a self-correspondence we get that $\underline{a} = \underline{b}$ and hence $\underline{a}\ \rho\ \underline{a}$. Furthermore, we obtain that $\underline{a}\ \rho^T\ \underline{a}$ and because k extends ρ^T we get that $T_{op}(\underline{a})\ \rho^T\ T_{op}(\underline{a})$. Using the main property of ρ^T as described above we get that $T_{op}(\underline{a})\ \rho\ T_{op}(\underline{a})$ and also that $T_{op}(\underline{a})$ is untagged. Now, notice that the standard value function induces a vertical correspondence ρ_0: $AI \leftrightarrow T$, i.e. $\hat{t}\ \rho_0\ t$. This means that $A_{op}(\underline{a})\ \rho_0\ T_{op}(\underline{a})$ and moreover because $T_{op}(\underline{a})$ is untagged we get that its standard value is equal to itself, or in other words that $A_{op}(\underline{a}) = T_{op}(\underline{a})$ which finally leads to $A_{op}(\underline{a})\ \rho\ A_{op}(\underline{a})$. □

Preservation of self-correspondences improves the quality of the properties that can be inferred from vertical globality. We no longer have to put all the loose elements into a relation in order to guarantee preservation, and therefore we can hope for a better definability result. Recall that loose definability comes with defining terms that can contain an arbitrary number of loose elements and a consequence of that is the loss of any usefulness when applied to operations that have a loose result sort. However, we have seen that in the case of iso-closed specifications it is enough to require only the reachable visible elements to be related and in accordance to that we can change first the definability definition, and then prove that we can find defining terms, for operations implemented via vertical global constructions, that depend only on those reachable visible elements.

Definition 4.22 (Visible definability)**.** *Let $\Sigma_{\mathbf{COL}}$ and $\Sigma'_{\mathbf{COL}}$ be two* **COL***-signatures and let $\sigma_{\mathbf{COL}}$ be a tight* **COL***-signature morphism between them. Let $op' : s_1, \ldots, s_n \to s \in OP'$ be an operation symbol and A' be a $\Sigma'_{\mathbf{COL}}$-algebra. We say that op' is* visible definable *on A' if for every tuple $a_1, \ldots, a_n$, with $a_i \in A'_{s_i}$ for $i = 1 \ldots n$, there exists a term $t \in T_{\Sigma \cup V^{a_1,\ldots,a_n}}(\{x_1, \ldots, x_n\})$ such that:*

$$A'_{op'}(a_1, \ldots, a_n) = A'_t[a_1/x_1, \ldots, a_n/x_n]$$

where $V^{a_1,\ldots,a_n}$ is the indexed set of reachable visible elements from $a_1, \ldots, a_n$, i.e. $V_v^{a_1,\ldots,a_n} = \langle Gen^{a_1,\ldots,a_n}_{\Sigma_{\mathbf{COL}}}(A)\rangle_v$ for all $v \in S_{Obs}$. The definition extends as before to sets of operations and classes of algebras and constructions.

Now we can prove that globality implies visible definability, i.e. defining terms for operations might depend only on a set of reachable visible elements, and need no longer be dependent on elements of hidden loose sorts.

Proposition 4.23 (Globality implies visible definability)**.** *Let* $k : SPI_{\mathbf{COL}} \Rightarrow SP_{\mathbf{COL}}$ *be a tight* **COL***-construction such that* $SPI_{\mathbf{COL}}$ *is iso-closed. If k is a vertically global construction then k is visible definable.*

Proof. Let A be a **COL**-algebra in $k(SPI_{\mathbf{COL}})$, i.e. there exists $AI \in SPI_{\mathbf{COL}}$ such that $k(AI) = A$. Let $op : w \to s \in OP$ and a tuple $\underline{a} \in A_w$. We define the self-correspondence ρ: $AI \leftrightarrow AI$ that relates $AI_t[\underline{a}]\ \rho\ AI_t[\underline{a}]$ for all $t \in T_{\Sigma \cup V^{\underline{a}}}(\{\underline{x}\})$ where $V^{\underline{a}}$ is the set of reachable visible elements from $\underline{a}$. Notice that this relation trivially satisfies the first requirement of self-correspondences, and by including $V^{\underline{a}}$ in it we make sure the second requirement is also satisfied. Now, because k extends self-correspondences we get that $A_{op}(\underline{a})\ \rho\ A_{op}(\underline{a})$ and therefore there exists a term t such that $A_{op}(\underline{a}) = AI_t[\underline{a}]$ □

It is not to be expected that vertical globality is equivalent to some sort of uniform visible definability. For uniformity one needs preservation of relations between different algebras, but self-correspondences do not offer that, they are just predicates.

4.3.1 Theorems for free

We have already used loose definability to prove properties for operations that have a constrained loose sort. The distinctive value of visible definability is shown when we try to obtain properties for generically implemented operations that have a loose result sort. For that we will use a coalgebraic specification of streams as a loose sort.

spec STREAM
  sorts *Stream*, *Elem*
  operations *hd* : *Stream* → *Elem*
    tl : *Stream* → *Stream*
  observers *hd*
  axioms
end

spec CUTSTREAM = STREAM **THEN**
  operations $cut : Stream \to Stream$
end

Consider a vertical global construction k : STREAM⇒CUTSTREAM. We will not repeat the scenario from previous examples by looking at the properties of its canonical lifting in a particular global context but rather we will try to explain why the name of the implemented function reflects its behavior. The property we are looking for is that a generically implemented *cut* operation does in fact "cut" a finite number of elements from the head of the stream.

Applying loose definability for *cut* leads us to an unsatisfactory result. We would get that the defining terms can contain any loose elements, that is arbitrary streams. Such a property does not restrict the results that can be produced by *cut* and that is why for obtaining a meaningful property we will make use of visible definability.

Visible definability applied to *cut* means that defining terms are in fact formed only from hd, tl and visible elements. But, because we have no way to build streams from visible elements we obtain that the defining terms are formed only from tl. Please notice that we cannot put in correspondence any two streams, and hence we cannot apply loose definability to obtain that the amount of cutting is the same for all elements and across algebras. Hence, it is not the case that generic constructions between STREAM and CUTSTREAM are in one to one correspondence with natural numbers, but they definitely perform a cutting operation.

4.4 Comparing general and vertical globality

We have seen until now, in Chapter 3 and Section 4.1, two kinds of global constructions. We will summarize the main results in the table below.

Vertical global constructions are more convenient when we want to base an argument on the preservation of a correspondence because we know for sure that any vertical correspondence is extended, and we don't need to show additionally that the correspondence is closed. Also, it is easier to integrate lifted constructions into vertical global contexts (see the differences between Proposition 3.4 and Proposition 4.2). In favor of general global constructions stands, first of all, their generality, i.e. the ability to use them in any context. That comes with more powerful definability results as only

the operations from a signature appear in the defining terms, in contrast to the vertical case for which defining terms may contain a finite number of elements from loose carriers.

Global constructions	General (Definition 3.3)	Vertical (Definition 4.1)
Correspondences	General correspondences: algebraic relations (Definition 3.5)	Vertical correspondences: algebraic relations bi-injective on visible sorts bi-surjective on loose sorts (Definition 4.3)
Globality equivalent to closed-stability	Yes. General globality is equivalent to closed general stability (Theorem 3.9)	Yes. Vertical globality is equivalent to closed vertical stability (Theorem 4.8)
Globality equivalent to stability	No. Not all general correspondences are preserved by generally global constructions (Example 3.15)	No. (arbitrary classes). Yes. (iso-closed classes). Vertical globality is equivalent to vertical stability. (Theorem 4.17)
Definability (point-wise)	Yes General globality implies algebraic definability (Proposition 3.11)	Yes Vertical globality implies loose definability (Proposition 4.10) or visible definability (Proposition 4.23)
Definability (uniform)	No. Not all generally global constructions are uniformly general definable. (Example 3.15)	Yes. For iso-closed classes. Vertically global constructions are uniformly loose definable. (Theorem 4.19)

In the next section, we will look briefly at a kind of global construction that falls

between the two already presented cases and borrows good properties from both of them.

4.5 Quasi-vertical global contexts

In this section we will look at a kind of fitting morphism which we will call quasi-vertical signature morphisms. The two main points that we want to insist on are the characterizations of globality via stability and via uniform definability. Recall that because the general global constructions are forced to extend only the closed correspondences they are incompletely characterized by uniform definability (see Example 3.15). The vertical global constructions, studied in a previous section, are well-behaved from that point of view, i.e. they extend all correspondences and we can choose the same defining term for any two correspondent tuples of arguments. However, the definability results are not as strong as for the general case: we only have uniform loose definability, or visible definability in the best scenario. Our goal is to find a notion of globality that will inherit the best properties from the already discussed cases. The way to do that is by using a notion of fitting signature morphism that falls between the notions of general signature morphism and vertical signature morphism. We will enumerate the corresponding definitions and the important properties for quasi-vertical global constructions will be given sometimes without a complete proof, unless the proof is significantly different from that given in the previous cases.

We start by defining the quasi-vertical signature morphisms as the ones that preserve only the visible sorts.

Definition 4.24 (Quasi-vertical signature morphism)**.** *Let* $\Sigma_{\mathbf{COL}}$ *and* $\Sigma'_{\mathbf{COL}}$ *be two* **COL***-signatures and let* $\phi : \Sigma \to \Sigma'$ *be a signature morphism between their underlying signatures. We say that* ϕ *is a quasi-vertical signature morphism if it preserves visible sorts, i.e.* $\phi(S_{Obs}) \subseteq S'_{Obs}$.

Similar to the general case, a quasi-vertical lifting pushout is a lifting pushout in which the fitting morphism ϕ is a quasi-vertical morphism.

$$\begin{array}{ccc} \Sigma I'_{\mathbf{COL}} & \xrightarrow{\sigma'_{\mathbf{COL}}} & \Sigma'_{\mathbf{COL}} \\ \uparrow{\scriptstyle \phi} & & \uparrow{\scriptstyle \phi'} \\ \Sigma I_{\mathbf{COL}} & \xrightarrow[\sigma_{\mathbf{COL}}]{} & \Sigma_{\mathbf{COL}} \end{array}$$

Definition 4.25 (Quasi-vertical global construction)**.** *Let* $k : SPI_{\mathbf{COL}} \Rightarrow SP_{\mathbf{COL}}$ *be a persistent* **COL***-construction. We say that k is a* quasi-vertical global **COL**-construction *if for every quasi-vertical lifting pushout,* $\phi(k)$ *is a* **COL***-construction between* $\phi(SPI_{\mathbf{COL}})$ *and* $\phi'(SP_{\mathbf{COL}})$.

The lifting to an arbitrary context of a quasi-vertical global construction is done as in the general case, using reachable and fully-abstract algebras only in one of the necessary inclusions.

Proposition 4.26 (Lifting quasi-vertical global constructions)**.** *Let* $k : SPI_{\mathbf{COL}} \Rightarrow SP_{\mathbf{COL}}$ *be a quasi-vertical global* **COL***-construction, and two specifications* $SPI'_{\mathbf{COL}}$ *and* $SP'_{\mathbf{COL}}$ *forming a global context connected by the following quasi-vertical lifting pushout*

$$\begin{array}{ccc} \mathsf{Sig}[SPI'_{\mathbf{COL}}] & \xrightarrow{\sigma'_{\mathbf{COL}}} & \mathsf{Sig}[SP'_{\mathbf{COL}}] \\ \uparrow{\scriptstyle \phi} & & \uparrow{\scriptstyle \phi'} \\ \mathsf{Sig}[SPI_{\mathbf{COL}}] & \xrightarrow[\sigma_{\mathbf{COL}}]{} & \mathsf{Sig}[SP_{\mathbf{COL}}] \end{array}$$

If the following conditions are satisfied:

1. $SPI'_{\mathbf{COL}} \subseteq \phi(SPI_{\mathbf{COL}})$
2. $\phi(k)(SPI'_{\mathbf{COL}}) \subseteq SP'_{\mathbf{COL}}$

then $k' : SPI'_{\mathbf{COL}} \Rightarrow SP'_{\mathbf{COL}}$*, defined as* $k'(A') = \phi(k)(A')$ *for each* $A' \in \mathsf{Mod}[SPI'_{\mathbf{COL}}]$*, is a* **COL***-construction. Moreover, the following conditions represent an equivalent version of the ones above, but they are more appropriate for practical use because they require the inclusion proofs to be done for a smaller class of algebras.*

1. $SPI'_{\mathbf{COL}} \subseteq \phi(SPI_{\mathbf{COL}})$
2. $\phi(k)(RFA(SPI'_{\mathbf{COL}})) \subseteq SP'_{\mathbf{COL}}$

The notion of correspondence must be also tuned for quasi-vertical usage.

Definition 4.27 (Quasi-vertical correspondence)**.** *Let* $\Sigma_{\mathbf{COL}} = (S, OP, OP_{Cons}, OP_{Obs})$ *be a* **COL***-signature and* A, B *be two* $\Sigma_{\mathbf{COL}}$*-algebras.* A (quasi-vertical) correspondence ρ *between A and B, written* $\rho : A \leftrightarrow B$*, is an algebraic relation between A and B that in addition is bi-injective on visible sorts.*

The definition of quasi-vertical stability is, as expected, the preservation of quasi-vertical correspondences.

Definition 4.28 (Quasi-vertical stable **COL**-construction)**.** *Let* $k : SPI_{\mathbf{COL}} \Rightarrow SP_{\mathbf{COL}}$ *be a tight* **COL***-construction. We say that k is* quasi-vertically (closed) stable *if it extends all quasi-vertical (closed) correspondences.*

We can prove that closed-stability characterizes globality using an almost identical procedure to the one used in the general case.

Theorem 4.29 (Globality is equivalent to closed-stability)**.** *Let* $k : SPI_{\mathbf{COL}} \Rightarrow SP_{\mathbf{COL}}$ *be a tight* **COL***-construction. Then k is quasi-vertically global if and only if it is quasi-vertically closed-stable.*

Proof. The proof is similar to that of Theorem 4.8.

Proving that quasi-vertical (closed) stability is sufficient in order to obtain quasi-vertical globality can be done by noticing that all isomorphisms in quasi-vertical fitted contexts are closed quasi-vertical correspondences (see Proposition 4.6 as the proof goes along the same lines).

The necessity of closed quasi-vertical stability follows from the fact that each closed quasi-vertical correspondence can be seen as a **COL**-isomorphism in a carefully chosen quasi-vertical context. The quasi-vertical context that is associated to a closed quasi-vertical correspondence is built similarly to how the associated contexts are built in the general and vertical cases (see Proposition 3.8 and Proposition 4.7). Basically the quasi-vertical case is a combination between the general case and the vertical case. The associated context for a quasi-vertical correspondence ρ has for every pair of related elements $a \; \rho \; b$ an explicit constructor constant $!^{a,b}$ (as in the general case), and an explicit observer $?^{b}$ only for state sorts (as in the vertical case). □

In the quasi-vertical case we can prove that all quasi-vertical correspondences should be preserved. That result is similar to the one proved for vertical global constructions as it requires that the source specification is iso-closed. In order to prove it we will show that a quasi-vertical correspondence can be decomposed into two quasi-vertical closed correspondences, via a helper algebra which is isomorphic to one of the related algebras. This is different than in the vertical case (Proposition 4.15) in which decomposition was symmetric, that is the helper algebra was isomorphic with both correspondent algebras. That happens basically because two algebras that are quasi-vertical correspondent are not necessarily isomorphic.

Proposition 4.30 (Decomposition of quasi-vertical correspondences)**.** *Let* $\Sigma_{\mathbf{COL}}$ *be a* **COL***-signature,* A, B *be two* $\Sigma_{\mathbf{COL}}$*-algebras and* ρ *be a quasi-vertical correspondence*

between them, i.e. $\rho: A \leftrightarrow B$*. Then there exists a* **COL***-algebra* A^ρ *such that* $A \equiv_{\Sigma_{\mathbf{COL}}} A^\rho$*, and two quasi-vertical closed correspondences* $\rho_A: A \leftrightarrow A^\rho$ *and* $\rho_B: A^\rho \leftrightarrow B$ *such that* $\rho_A;\rho_B{=}\rho$.

Proof. The idea behind the definition of A^ρ is to slightly modify A by adding some copies for the elements that are related by ρ. More concretely, for each sort $s \in S$, A^ρ_s will contain all the elements $a \in A_s$ that do not appear in ρ_s, and also all the pairs from ρ_s, i.e. $(a,b) \in \rho_s$ implies $(a,b) \in A^\rho_s$. We can define the A-value $\hat{m}$ for an element $m \in A^\rho$ by letting $\hat{m} = a$ if $m = a$ or $m = (a,b)$. To sum up, the carriers of A^ρ have pairs from ρ, and singletons which are not related by ρ. The operations are defined in order to respect that distinction and also to commute with the pairs. Let $op : w \to s$ and let $\underline{m} \in A^\rho_w$. We first apply op to the A-values of $\underline{m}$ and for that let $A_{op}(\underline{\hat{m}}) = a$. If a is unrelated by ρ then the result in A^ρ will be a singleton, i.e. $A^\rho_{op}(\underline{m}) = a$. Otherwise, when there exists $b \in B$ such that $a\ \rho\ b$, we treat two cases. The first case is when one of the arguments in $\underline{m}$ is a singleton and then we can choose any pair that contains a as a result for $A^\rho_{op}(\underline{m})$. The second case, when all the arguments in $\underline{m}$ are pairs, we set the result by evaluating the tuples that compose $\underline{m}$, i.e. $A^\rho_{op}(\underline{m}) = (A_{op}(\underline{a}), A_{op}(\underline{b}))$. So, the operations basically preserve ρ whenever it is possible.

To see that A^ρ is an isomorphic **COL**-algebra to A we have to notice that the characteristics of elements from A^ρ are similar to their A-values. The reachability constraints can be explained by checking that $m \in Gen_{\Sigma_{\mathbf{COL}}}(A^\rho)$ if $\hat{m} \in Gen_{\Sigma_{\mathbf{COL}}}(A)$, and also that $a \in Gen_{\Sigma_{\mathbf{COL}}}(A)$ implies that there exists $m \in Gen_{\Sigma_{\mathbf{COL}}}(A^\rho)$ such that $\hat{m} = a$. The observability equality comes from A, i.e. $m_0 \approx_{\Sigma_{\mathbf{COL}},A^\rho} m_1$ if and only if $\hat{m}_0 \approx_{\Sigma_{\mathbf{COL}},A^\rho} \hat{m}_1$. These facts makes it easy to check that $A^\rho \equiv_{\Sigma_{\mathbf{COL}}} A$.

We can now define the relations $\rho_A: A \leftrightarrow A^\rho$ and $\rho_B: A^\rho \leftrightarrow B$ as $a\ \rho_A\ (a,b)$ and $(a,b)\ \rho_B\ b$ for all $(a,b) \in \rho$. These are closed quasi-correspondences and $\rho{=}\rho_A;\rho_B$. □

Please notice that the decomposition property is different from the one proved in the vertical case (see Proposition 4.15). Typically, in the quasi-vertical case we cannot make the decomposition producing an intermediate algebra that is isomorphic to both related algebras, and hence Proposition 4.15 cannot be reused.

We can show under the requirement that the source specification is iso-closed that quasi-vertical globality is equivalent to quasi-vertical stability. As one implication is immediate from Theorem 4.29, we only have to show that globality implies stability, i.e. global constructions extend *all* correspondences.

Theorem 4.31 (Globality is equivalent to stability)**.** *Let* $k : SPI_{\mathbf{COL}} \Rightarrow SP_{\mathbf{COL}}$ *be a tight* **COL***-construction such that* $SPI_{\mathbf{COL}}$ *is iso-closed. Then k is quasi-vertically global if and only if it is quasi-vertically stable.*

Proof. As stability is stronger than closed-stability we can use Theorem 4.29 to discard one direction of the proof obligation. In order to show that globality implies preservation of all correspondences, we consider a quasi-vertical correspondence $\rho: AI \leftrightarrow BI$. Using the decomposition property (Proposition 4.30) we get that $\rho = \rho_{AI}; \rho_{BI}$ where $\rho_{AI}: AI \leftrightarrow AI^{\rho}$ and $\rho_{BI}: AI^{\rho} \leftrightarrow BI$ are closed quasi-vertical correspondences and $AI^{\rho} \in SPI_{\mathbf{COL}}$. Finally we use closed-stability for k and obtain that ρ is also extended. □

Now we can turn our attention to definability which is an immediate result using the same proof as for the general case.

Proposition 4.32 (Globality implies definability)**.** *Let* $k : SPI_{\mathbf{COL}} \Rightarrow SP_{\mathbf{COL}}$ *be a tight* **COL***-construction. If k is quasi-vertically global then k is definable.*

Proof. This proof is exactly the same as in the general case (see proof of Proposition 3.11). More explicitly, in order to show that an operation applied to a set of arguments produces a result that is term generated from its arguments we use the minimal quasi-vertical correspondence that contains those arguments. □

However, unlike for the general case, we can prove that defining terms for quasi-vertical correspondent arguments must be the same. That happens because all correspondences are preserved and the notion capturing that property is defined as follows:

Definition 4.33 (Uniform quasi-vertical definability)**.** *Let* $\Sigma_{\mathbf{COL}}$ *and* $\Sigma'_{\mathbf{COL}}$ *be two* **COL***-signatures and let* $\sigma_{\mathbf{COL}}$ *be a tight* **COL***-signature morphism* $\sigma_{\mathbf{COL}} : \Sigma_{\mathbf{COL}} \to \Sigma'_{\mathbf{COL}}$*. We say that an operation* $op' : w \to s \in OP'$ *is* uniformly quasi-vertical definable *on a class* $\mathcal{A}$ *of* $\Sigma_{\mathbf{COL}}$*-algebras if for all* $A, B \in \mathcal{A}$*, all quasi-correspondences* $\rho: A \leftrightarrow B$ *and all* ρ*-correspondent tuples* $\underline{a} \in A_w$ *and* $\underline{b} \in B_w$ *there exists a common defining* Σ*-term t for* op' *on* $\underline{a}$ *and on* $\underline{b}$*.*

We can easily show that uniform quasi-vertical definability implies quasi-vertical stability using the same reasoning as in Proposition 3.13.

Proposition 4.34 (Uniform definability implies stability)**.** *Let* $k : SPI_{\mathbf{COL}} \Rightarrow SP_{\mathbf{COL}}$ *be a tight* **COL***-construction. If k is uniformly quasi-vertical definable then k is quasi-vertically stable.*

Proof. The proof follows by an argument that is similar to the one used for proving Proposition 3.13 for the general case. □

Now we can prove that uniform definability is necessary for global quasi-vertical constructions.

Theorem 4.35 (Globality is equivalent to uniform definability)**.** *Let $k : SPI_{\mathbf{COL}} \Rightarrow SP_{\mathbf{COL}}$ be a tight* **COL***-construction such that $SPI_{\mathbf{COL}}$ is iso-closed. Then k is quasi-vertically global if and only if k is uniformly quasi-vertical definable.*

Proof. The fact that uniform definability is sufficient for obtaining globality follows by an argument that is similar to the one used for proving Proposition 3.13.

The sufficiency is proved by using stability which is a consequence for global constructions on iso-closed classes. For that let AI, BI be two $\Sigma I_{\mathbf{COL}}$-algebras, let $op : w \to s \in OP$ and let $\underline{a} \in AI_w$ and $\underline{b} \in BI_w$ be two correspondent elements via a quasi-vertical correspondence ρ: $AI \leftrightarrow BI$. We will denote by A and B the constructed algebras $k(AI)$ and resp. $k(BI)$. We define another quasi-vertical correspondence ρ_0: $AI \leftrightarrow BI$ that relates $AI_t[\underline{a}]\ \rho_0\ BI_t[\underline{b}]$ for all $t \in T_\Sigma(\{\underline{x}\})$. Because k extends ρ_0 we have that $A_{op}(\underline{a})\ \rho_0\ B_{op}(\underline{b})$and we can conclude that there exists a term t that defines op for $\underline{a}$ and resp. $\underline{b}$. □

4.5.1 Theorems for free

The characteristics of quasi-vertical globality are very similar to the ones of general globality. The distinctive difference is the uniform definability result, that allows us to conclude that implemented operations are defined by the same term when applied to related arguments. To see how that improves the reasoning we will present an example in which we have a local context with no observers with the aim to implement a merging function $merge : List, List \to Container$.

spec LISTSKELETON
  sorts *List*, *Elem*
  operations *insert*, *remove* : *Elem*, *List* → *List*
  hidden *List*
end

spec LISTSKELETONWITHMERGE = LISTSKELETON **THEN**
sorts $List, Elem$
operations $merge : List, List \rightarrow List$
end

We will look at properties of a lifting to the following quasi-vertically fitted context:

spec LISTWITHNAME
sorts $List, Elem$
operations $empty : String \rightarrow List$
$insert, remove : Elem, List \rightarrow List$
$name : List \rightarrow String$
constructors $empty, insert$
observers $(name)$
axioms
$name(empty(s)) = s$
$name(insert(x,c)) = name(c)$
$name(remove(x,c)) = name(c)$
end

spec LISTWITHNAMEANDMERGE = LISTWITHNAME **THEN**
operations $merge : List, List \rightarrow List$
axioms **// This is a theorem for free**
$\forall x, y.name(merge(x,y)) = name(x)$
$\vee$
$\forall x, y.name(merge(x,y)) = name(y)$
end

Please notice that the fitting morphism is quasi-vertical but it is not vertical. That happens because *List* is a hidden sort both in the local context and in the global context, but it is loose locally and constructed globally. Also, notice that the reducts of the models from the global specifications are **COL**-algebras for the signatures of the local

specifications. That is not always the case for reducts along quasi-vertical signature morphisms, but because we do not have any functions of visible result sort in the local context all standard algebras are **COL**-algebras.

Consider k to be a quasi-vertical construction in the local context. We get that *merge* is uniformly definable. Please notice, that by not having any function from the state sort *List* to the visible sort *Bool* we can easily conclude that all inhabitants of sort *List* are relatable via a quasi-vertical correspondence. After these remarks we can say that uniformity means that for any tuple of arguments we can find a common term that defines *merge* when applied to them. Uniform definability implies that *merge* acts like a chosen projection for all possible arguments, i.e. it cannot choose to project the name of the first argument or the name of the second argument based on an examination of the arguments. The free theorem for this case will be

$$(\forall x, y.name(merge(x,y)) = name(x)) \vee (\forall x, y.name(merge(x,y)) = name(y))$$

which is stronger than the one inferable just by using plain definability:

$$\forall x, y.name(merge(x,y)) = name(x) \vee name(merge(x,y)) = name(y)$$

4.6 Summary

In this chapter we have investigated the properties of global constructions depending on the restrictions that are made to the usage contexts. We have identified how the notion of correspondence must change in each case to ensure a sound and complete characterization of globality. The preservation of loose sorts in the lifting process is captured by the bi-surjectivity of the characterizing relation on the loose sorts, while the preservation of visible sorts is captured by the bi-injectivity on the visible sorts. Unlike the general case presented in Chapter 3, both vertical and quasi-vertical global constructions can be characterized by means of stability, rather than just closed-stability. These results are in the same tone as those proved in [Sch87, BST08].

We have also proved definability results for both cases. The definability result for vertical global constructions is rather poor at first as it states that defining terms can contain an arbitrary number of elements from the loose carriers, but in the end we show how it can be strengthened to allow only elements from the visible sorts in them. The visible elements that might appear in defining terms for vertically global constructions are those that are generated from elements of loose sorts. That is because the promise

is that in global contexts no element of the loose sort will be discarded, so their visible outcome cannot be ignored.

The definability notion for the quasi-vertical case is algebraic definability, the difference compared to the general case being that uniform definability is obtained as well.

An updated comparison table describing the features of the three cases of global constructions that have been investigated so far is given below.

Global constructions	General (Definition 3.3)	Quasi-vertical (Definition 4.25)	Vertical (Definition 4.1)
Correspondences	algebraic relations (Definition 3.5)	bi-injective on *Obs* (Definition 4.27)	bi-injective on *Obs* bi-surjective on *Loose* (Definition 4.3)
Globality equivalent to closed-stability	Yes. General globality is equivalent to closed general stability (Theorem 3.9)	Yes. Quasi-vertical globality is equivalent to closed quasi-vertical stability. (Theorem 4.29)	Vertical globality is equivalent to closed vertical stability. (Theorem 4.8)
Globality equivalent to stability	No. General globality does not imply general stability (Example 3.15)	Yes (iso-closed classes). Quasi-vertical globality is equivalent to quasi-vertical stability (Theorem 4.31)	Yes (iso-closed classes). Vertical globality is equivalent to vertical stability. (Theorem 4.17)
Definability (point-wise)	Yes General globality implies algebraic definability. (Proposition 3.11)	Yes Quasi-vertical globality implies algebraic definability. (Proposition 4.32)	Yes. Vertical globality implies loose definability. (Prop. 4.10) or visible definability (Proposition 4.23)
Definability (uniform)	No General globality does not imply uniform algebraic definability. (Example 3.15)	Yes (iso-closed). Quasi-vertical globality is equivalent to uniform algebraic definability. (Theorem 4.35)	Yes (iso-closed). Vertical globality is equivalent to uniform loose definability. (Theorem 4.19)

Chapter 5

Constructions for higher order types

Contents

In this chapter we will analyze the features of constructions for which the implemented functions are denoted by functional constants on higher order sorts. We do that primarily in order to mirror the setting of lambda calculus and to be able to obtain "theorems for free" for higher order types. However, our algebraic approach will reveal new facets of the abstraction barriers imposed by stability and we will analyze these thoroughly.

5.1 Introduction

As we have said previously in Chapter 3 one of the informal guidelines in the investigation of properties induced by stable constructions is the similarity between stability and the notion of parametricity present in the framework of second order lambda calculus. This suggested to us the use of the catch phrase "theorems for free", originally used by Wadler in [Wad89] to describe properties inferred from parametricity, in the algebraic specifications world. Some "theorems for free" for stable constructions were

presented in Chapter 3 and in Chapter 4. In those chapters we showed that just by looking at the source and target signatures of a construction we can extract properties for that construction without knowing its actual implementation. However, the substance of "theorems for free" is the link between the type of an implemented function and some property it should satisfy. In the playful words of Wadler, the aim is the following:

> Write down the definition of a polymorphic function on a piece of paper. Tell me its type, but be careful not to let me see the function's definition. I will tell you a theorem that the function satisfies.

While this is to some extent captured by the work presented in the previous chapters, if we allow ourselves to think of the signatures of a construction as defining its type, we shall look in this chapter at a more direct fit of the concepts from System F. We will work with type hierarchies inside signatures in order to express properties about generically implemented functions just by looking at their types.

First, in Section 5.2 we will present the basic setting for working with higher order types in **COL** signatures. Then, in Section 5.3, we will look at how the global constructions, that have been already been investigated in the previous chapters, interact with the richer type hierarchies. In Section 5.4, we will investigate one of the features of function types, extensionality, in relation with the notion of global construction. The global constructions that can be used in extensional contexts will turn out to be characterized by logical relations, and hence they will be named *logical global constructions*. At that point we will be able to note a closer similarity than before between logical stability and parametricity, which is defined in System F as preservation of logical relations. We will see that logical global constructions produce operations that are not necessarily definable in terms of the source signature. In the last section, Section 5.5, we will discuss some problems concerning representation independence; among other things we will give a sound and complete method for showing observational equivalence of **COL**-algebras based on logical relations.

5.2 Higher order sorts in COL

Higher order types can be represented in **COL** in a straightforward manner starting from standard simply typed signatures. In fact, the observational aspects provided by **COL** allow for a smoother representation of extensionality than in the standard algebraic framework. The typical way to represent extensional higher order types is to

constrain the models by an axiom, i.e. the extensionality axiom. In **COL** this constraint can be represented at the signature level by requiring that the application operation is the only observer for higher order types. Informally, that can be understood immediately if we realize that extensional equality is an observational equality; it is a relation between functions that cannot be distinguished through application and extensional higher order types are just particular examples of hidden sorts. We will express the basic notions in the following definitions.

Definition 5.1 (Functional signatures in **COL**)**.** *A signature* $\Sigma_{\mathbf{COL}}$ *is* simply typed *if the underlying signature* Σ *is simply typed. If in addition* Σ *is combinatorial we call* $\Sigma_{\mathbf{COL}}$ *combinatorial too.*

A simply typed signature $\Sigma_{\mathbf{COL}}$ *is called* applicative *if the application operations are among the observers on the higher order types, i.e.* $APP_S \subseteq OP_{Obs}$.

An applicative signature $\Sigma_{\mathbf{COL}}$ *is called* functional *if the application operations are the only observers on the higher order types, i.e. for all* $s_0 \Rightarrow s_1 \in S$ *we have that* $OP_{Obs}^{s_0 \Rightarrow s_1} = \{(app_{s_0,s_1}, 1)\}$.

Notice that standard simply typed signatures can be written in a canonical way by currying all the operation symbols, except those corresponding to application, and presenting them as constants of higher order types. However, the semantics of **COL** constraints means that it is significant whether a function is represented as a constant or as an operation. In particular, observers cannot be constant symbols; they must have at least one argument sort, i.e. the observed sort. So, we cannot present **COL** signatures in a canonical way, but we should not regard that as a disadvantage. There is a useful distinction that comes from allowing more than one hierarchy of types. In simply typed **COL**-signatures there are two type hierarchies; an implicit hierarchy of types, that is not represented in the carrier sets but rather as functions between the carrier sets, used for giving meaning to the operations present in the signature; and an explicit hierarchy of types which is used to give meaning to functional constants. Methodological advice is to treat as constants of a higher order sort only those functions which rightfully belong to the intended semantics, i.e. over which we intend to quantify in our axioms. We will see how the existence of two type hierarchies impacts the process of writing implementations (see Section 5.3.1). Also, this separation will be crucial in our attempt to characterize globality in terms of stability in Theorem 5.15.

Please note, that even if observers and constructors are not required to be present in the carriers of the algebras, there might be cases when we would like them to be

represented somehow in the carriers. That can easily be obtained by putting in the signatures some higher order constants that behave like the corresponding uncurried operations. In order to simplify notation we will introduce a new syntax for specifications (**hspec** SPECNAME) that have all their uncurried operations represented as curried higher order constants with the behavior of those constants enforced by axioms. For example the following specification:

hspec BRIEFSPEC **then**
  sorts s, v
  operations $obs : s \to v$
    $empty : s$
    $cons : s, v \to s$
  combinatorial
  constructors $empty, cons$
  observers obs
end

is equivalent to a standard **COL** specification that has two additional constant obs' and $cons'$ and some axioms defining their behavior.

spec FULLSPEC **then**
  sorts s, v
  operations $obs : s \to v$
    $empty : s$
    $cons : s, v \to s$
    $obs' : s \Rightarrow v$
    $cons' : s \Rightarrow v \Rightarrow s$
  combinatorial
  constructors $empty, cons$
  observers obs
  axioms $app(obs', x) = obs(x)$
    $app(app(cons', x), y) = cons(x, y)$
end

We do that in order to write lambda expressions in BRIEFSPEC (using the $\mathcal{S}, \mathcal{K}$-combinators) that contain all the symbols of the signature. For example we write $\lambda x, v.cons(cons(x, v), v)$, because we understand it as being a combinatorial term using the higher order constant $cons'$ instead of $cons$. Before getting any further please

have in mind that we are not enriching **COL**, and that we are using this new notation just for the sake of brevity in cases when we would like the operations of the signature to have a representative value in the carrier of the algebras.

5.2.1 Extensionality in COL

Our claim from the beginning of this section was that extensionality is captured by the constraints induced by a particular kind of simply typed **COL** signatures, namely by functional signatures, i.e. those for which the only observer on higher order sorts is application. We will show that algebras of functional **COL**-signatures are always extensional. The converse, i.e. that extensional **COL**-algebras can always be seen as algebras of a canonical functional signature, is not necessarily true for **COL** signatures but we will identify a class of simply typed **COL** signatures for which that holds (see Proposition 5.8).

Extensional **COL**-algebras are those that satisfy the extensionality axioms w.r.t. **COL**-satisfaction.

Definition 5.2 (Extensional **COL**-algebras)**.** *Let $\Sigma_{\mathbf{COL}}$ be a simply typed* **COL***-signature. A $\Sigma_{\mathbf{COL}}$-algebra A is called* extensional *if $A \models_{\Sigma_{\mathbf{COL}}} ExtAx_{\Sigma}$.*

A property equivalent to extensionality can be expressed by characterizing the observational equality of an extensional **COL**-algebra.

Proposition 5.3. *Let $\Sigma_{\mathbf{COL}}$ be a simply typed* **COL***-signature. A $\Sigma_{\mathbf{COL}}$-algebra A is extensional if and only if $\approx_{\Sigma_{\mathbf{COL}},A}$ is a logical relation on $\langle Gen_{\Sigma_{\mathbf{COL}}}(A)\rangle$.*

Proof. Let A be $\Sigma_{\mathbf{COL}}$-algebra.

Assume that A is extensional, i.e. $A \models_{\Sigma_{\mathbf{COL}}} ExtAx_{\Sigma}$. Let $f,g \in \langle Gen_{\Sigma_{\mathbf{COL}}}(A)\rangle_{s_0 \Rightarrow s_1}$ such that for all $a,b \in \langle Gen_{\Sigma_{\mathbf{COL}}}(A)\rangle_{s_0}$ with $a \approx_{\Sigma_{\mathbf{COL}},A} b$ we have $f(a) \approx_{\Sigma_{\mathbf{COL}},A} g(b)$. This means that for all $a \in \langle Gen_{\Sigma_{\mathbf{COL}}}(A)\rangle_{s_0}$ we have $f(a) \approx_{\Sigma_{\mathbf{COL}},A} g(a)$ which by extensionality implies $f \approx_{\Sigma_{\mathbf{COL}},A} g$.

Now assume that $\approx_{\Sigma_{\mathbf{COL}},A}$ is logical on $\langle Gen_{\Sigma_{\mathbf{COL}}}(A)\rangle$ and that for some $f,g \in \langle Gen_{\Sigma_{\mathbf{COL}}}(A)\rangle_{s_0 \Rightarrow s_1}$, $A \models_{\Sigma_{\mathbf{COL}}} \forall x. f(x) = g(x)$. Using that and the fact that $\approx_{\Sigma_{\mathbf{COL}},A}$ is a congruence we have for all $a,b \in \langle Gen_{\Sigma_{\mathbf{COL}}}(A)\rangle_{s_0}$ with $a \approx_{\Sigma_{\mathbf{COL}},A} b$ that $f(a) \approx_{\Sigma_{\mathbf{COL}},A} g(a) \approx_{\Sigma_{\mathbf{COL}},A} g(b)$. Which in the end proves $f \approx_{\Sigma_{\mathbf{COL}},A} g$ because $\approx_{\Sigma_{\mathbf{COL}},A}$ is logical. □

Proposition 5.4 (Functional algebras are extensional)**.** *Let $\Sigma_{\mathbf{COL}}$ be a functional* **COL***-signature and A be a $\Sigma_{\mathbf{COL}}$-algebra. Then A is extensional.*

Proof. Let A be a $\Sigma_{\mathbf{COL}}$-algebra and let $s_0 \Rightarrow s_1 \in S$ and $f, g \in \langle Gen_{\Sigma_{\mathbf{COL}}}(A)\rangle_{s_0 \Rightarrow s_1}$ such that for all $a \in \langle Gen_{\Sigma_{\mathbf{COL}}}(A)\rangle_{s_0}$ we have $f(a) \approx_{\Sigma_{\mathbf{COL}},A} g(a)$. This leads to $f \approx_{\Sigma_{\mathbf{COL}},A} g$ because *app* is the only observer on $s_0 \Rightarrow s_1$. Therefore, $\approx_{\Sigma_{\mathbf{COL}},A}$ is logical on $\langle Gen_{\Sigma_{\mathbf{COL}}}(A)\rangle$ and consequently extensional. □

A question that remains to be answered for the sake of completeness is if all extensional **COL**-algebras can be represented as algebras of a canonical functional **COL**-signature. This is not a crucial result so the rest of this section can be skipped by a selective reader. The important conclusion is that we will use functional signatures for representing extensionality in **COL**.

We will see in the following example that it is not always the case that extensional algebras can be represented as algebras of a canonical functional signature. We produce a counterexample in which we start with an extensional algebra of a simply typed **COL** signature and then by eliminating all the observers on the higher order sorts from the signature, except the applications, we end up altering the observational equality and the original algebra will not be an observational algebra of the newly obtained signature. Before presenting that counterexample we will see how we can associate to each simply typed **COL**-signature a functional **COL**-signature by forcing the *app* operations to be the only observers on higher order types.

Definition 5.5 (Associated functional signature)**.** *Let $\Sigma_{\mathbf{COL}} = (\Sigma, OP_{Cons}, OP_{Obs})$ be a simply typed* **COL***-signature. The signature $\Sigma^*_{\mathbf{COL}} = (\Sigma, OP_{Cons}, OP^*_{Obs})$ is called* the associated functional signature of $\Sigma_{\mathbf{COL}}$ *where OP^*_{Obs} contains the same observers for basic types as OP_{Obs} and only the application operations app as observers on higher order types.*

The following example presents an extensional algebra **COL** that cannot be seen as a **COL**-algebra for the associated functional signature.

Example 5.6 (Extensional but not functional)**.** *Let $\Sigma_{\mathbf{COL}}$ be the following* **COL** *signature:*

spec EXTNOFUN = BOOL ***then***
  sorts $s, s \Rightarrow s$
  operations $obs_0 : s \to (s \Rightarrow s)$
    $obs_1 : s \Rightarrow s \to Bool$
    $app : s \Rightarrow s, s \to s$
  observers $obs_0, obs_1, (app, 1)$
end

and let A be the following $\Sigma_{\mathbf{COL}}$*-algebra:*

$$A_s = \{a,b\},\ A_{s \Rightarrow s} = \{f,g\}$$

$$A_{obs_0} = [a \mapsto f, b \mapsto g]$$

$$A_{obs_1} = [f \mapsto \mathit{true}, g \mapsto \mathit{false}]$$

$$A_{app}(f,x) = a \text{ and } A_{app}(g,x) = b \text{ for all } x \in A_s$$

and let $\Sigma^*_{\mathbf{COL}}$ *be the associated functional signature, i.e.* $\Sigma^*_{\mathbf{COL}} = (\Sigma, \emptyset, OP_{Obs} \setminus \{obs_1\})$.

We claim that $A \models_{\Sigma_{\mathbf{COL}}} ExtAx_\Sigma$ *but it is not a* $\Sigma^*_{\mathbf{COL}}$*-algebra.*

Proof. First notice that $\approx_{\Sigma_{\mathbf{COL}},A}$ is the identity relation and hence A is a fully-abstract $\Sigma_{\mathbf{COL}}$-algebra. Furthermore it is obvious that A satisfies the extensionality axiom.

Now, if we try to view A as a functional algebra of $\Sigma^*_{\mathbf{COL}}$ we realize that $\approx_{\Sigma^*_{\mathbf{COL}},A}$ is the total relation. From this we can conclude that A is not a $\Sigma^*_{\mathbf{COL}}$-algebra because A_{obs_1} does not give identical results for the related arguments $f \approx_{\Sigma^*_{\mathbf{COL}},A} g$. □

The reason behind the previous counterexample is the presence of the observer obs_0 which makes the observational equality on the basic sort s dependent on that on the higher order sort $(s \Rightarrow s)$. We can recover a positive result by showing that for signatures that disallow such observers (see Definition 5.7 below) extensionality is synonymous with functionality.

Definition 5.7 (Straight observers)**.** *Let* $\Sigma_{\mathbf{COL}}$ *be a* **COL***-signature. An observer* $(obs, i) \in OP_{Obs}$*, where* $obs : s_1, \ldots, s_n \to s$*, is called* straight *if* s *is a basic type whenever* s_i *is a basic type.*

We can now prove that if observational equality on basic sorts depends only on other basic sorts, eliminating observers for higher order sorts does not modify it.

Proposition 5.8 (Straight extensional algebras are functional)**.** *Let* $\Sigma_{\mathbf{COL}}$ *be a simply typed* **COL***-signature with straight observers and let* $\Sigma^*_{\mathbf{COL}}$ *be its associated functional* **COL***-signature. Then any extensional* $\Sigma_{\mathbf{COL}}$*-algebra A is also a* $\Sigma^*_{\mathbf{COL}}$*-algebra.*

Proof. Let $\approx$ be the observational equality $\approx_{\Sigma_{\mathbf{COL}},A}$ w.r.t. $\Sigma_{\mathbf{COL}}$, and let $\approx^*$ be the observational equality $\approx_{\Sigma^*_{\mathbf{COL}},A}$ w.r.t. $\Sigma^*_{\mathbf{COL}}$.

As the signatures $\Sigma_{\mathbf{COL}}$ and $\Sigma^*_{\mathbf{COL}}$ differ only in the set of observers, we can say that the corresponding generated algebras coincide and it is sufficient to prove that $\approx = \approx^*$ in order to conclude that A is a $\Sigma^*_{\mathbf{COL}}$-algebra.

It is easy to see that $\approx \subseteq \approx^*$ because each observable context formed from $\Sigma^*_{\mathbf{COL}}$ can be seen as an observing term over $\Sigma_{\mathbf{COL}}$, that will respect $\approx$ and therefore will give equal results for $\approx$ related elements.

Now, we will prove by induction over the type structure that $\approx^* \subseteq \approx$. First, let s be a basic type, and let $a, b \in \langle Gen_{\Sigma^*_{\mathbf{COL}}}(A)\rangle_s$ be two elements such that $a \approx^*_s b$. Because the observers on basic types are straight, i.e. they do not lead to higher order type results, and the difference between the two signatures is just for observers of higher order types, we can see that $\mathcal{C}(\Sigma_{\mathbf{COL}})_{s\to v} = \mathcal{C}(\Sigma^*_{\mathbf{COL}})_{s\to v}$ for every observable sort $v \in S_{Obs}$. Therefore the observational equalities coincide on all basic types s. Secondly, let us consider a higher order type $s_0 \Rightarrow s_1$ and two related functions $f \approx^*_{s_0\Rightarrow s_1} g$ w.r.t. the associated signature. Because f and g are related we get that for all arguments $a \in \langle Gen_{\Sigma^*_{\mathbf{COL}}}(A)\rangle_{s_0}$ we have that $app(f,a) \approx^*_{s_1} app(g,a)$. Now, using the induction hypothesis, $\approx^*_{s_1} \subseteq \approx_{s_1}$, we get that $app(f,a) \approx_{s_1} app(g,a)$. And finally from the fact that A is extensional we get that $f \approx_{s_0\Rightarrow s_1} g$. □

We consider that these results that relate extensionality with functional signatures are enough to justify the following methodological rule: whenever we want to represent extensional higher order types we shall use functional signatures.

5.3 Implementation of functions as higher order constants

In Chapter 3 and Chapter 4 we have looked at constructions that implement functions represented as additional operation symbols. In this section we will focus on the differences that appear when we have an explicit type hierarchy present in our signatures and implemented functions are represented as additional constant symbols on some higher order types. We will present our findings using the quasi-vertical global constructions introduced in Section 4.5. We make this choice because quasi-vertical globality implies a simple kind of definability, namely algebraic definability, and also it is characterized by a relatively simple kind of correspondence, the quasi-vertical correspondences, i.e. algebraic relations which are bi-injective on visible sorts. However, this choice is not that important as the discussion that follows is easily translatable to the setting of general or vertical global constructions. To emphasize the generality of the development we will not use the "quasi-vertical" qualification unless it is needed to compare it to one of the other variants. So, the assumptions for this section are that

we are working with a quasi-vertical global construction $k : SPI_{\mathbf{COL}} \Rightarrow SP_{\mathbf{COL}}$ such that signatures $\Sigma I_{\mathbf{COL}}$ and $\Sigma_{\mathbf{COL}}$ are simply typed (see Definition 5.1) and the tight signature morphism $\sigma : \Sigma I \to \Sigma$ adds only constant symbols of the form $op : s_0 \Rightarrow s_1$ on some higher order type.

The lifting to global contexts is done via lifting pushouts like the one below where ϕ is an quasi-vertical signature morphism.

$$\begin{array}{ccc} \Sigma I'_{\mathbf{COL}} & \xrightarrow{\sigma'_{\mathbf{COL}}} & \Sigma'_{\mathbf{COL}} \\ \uparrow{\scriptstyle \phi} & & \uparrow{\scriptstyle \phi'} \\ \Sigma I_{\mathbf{COL}} & \xrightarrow[\sigma_{\mathbf{COL}}]{} & \Sigma_{\mathbf{COL}} \end{array}$$

This view entitles us to use the results that have already been proven in Chapter 4, but insisting on the fact that the implemented operations have a special characteristic, i.e. they are constant symbols.

For example, Proposition 4.32 ensures that global constructions add symbols that are definable. Notice however that there is an important difference between what definability means for functions represented as operations with a non-nullary arity, as in the examples of previous chapters, and functions represented as constant symbols, as we consider in this section. Algebraic definability (see Definition 3.10) means in the first case that there is a defining term for each set of arguments. In the second case, because we don't have argument sorts, it means that there is single defining term for the implemented function.

Now, we will look at an example that corresponds to a well known fact from System F: the only parametric inhabitants of the type $\forall X.(X \to X) \to X \to X$ are the Church numerals, i.e. $\lambda X.\lambda s.\lambda 0.s^n(0)$ for $n \in \mathbb{N}$ (see [Rey83, GLT89]). To mimic that in **COL** we will choose a construction that starts from a trivial combinatorial signature, i.e. one that has no operations other than the applications and the combinators, and that implements a function with a type corresponding to the one of the Church numerals. We want to prove that global constructions from EMPTY to CHURCH are in one to one correspondence with the standard implementations represented by Church numerals.

```
spec EMPTY
  sorts Elem
  combinatorial
end
```

spec CHURCH = EMPTY **then**
  operations $op : (Elem \Rightarrow Elem) \Rightarrow Elem \Rightarrow Elem$
end

Let k : EMPTY⇒CHURCH be a global construction. Because k is global we get that it is also definable and hence for each $AI \in$ EMPTY there exists a defining term t such that $A_{op} = AI_t$. Please notice that the defining terms are in fact combinatorial terms, i.e. terms formed only from applications of the $\mathcal{S}, \mathcal{K}$ combinators, because there are no other operations in the signature of EMPTY. In conclusion, for an algebra $AI \in$ EMPTY the operation op is defined by a combinatorial term. One can prove that each combinatorial term t of type $(Elem \Rightarrow Elem) \Rightarrow Elem \Rightarrow Elem$ is observationally equal to a Church numeral $\mathcal{C} : (Elem \Rightarrow Elem) \Rightarrow Elem \Rightarrow Elem$. Taking into account that k is also uniformly definable and all algebras of EMPTY can be related to each other through quasi-vertical correspondences we get that a unique defining term can be chosen for all algebras and hence each global construction corresponds to a standard implementation given by a Church numeral.

5.3.1 Mixing type hierarchies

We have seen that the treatment of functions as higher order constants allows us to get properties more in line with the ones presented originally for System F. However, persistent constructions that implement functions as constants on a higher order type have one big disadvantage. They must give an interpretation in the source algebra to the newly added functions, and hence they cannot add truly "new" functions. Moreover, global constructions require that the implementation of the newly added constant should be definable and hence one is able to express only what is term definable in the given signature. A consequence of that is that in the specification of a data type we should include all combinators that will be used for defining higher order functions. In particular, if we want to allow constructions that are recursively defined we must add an iterator or a fix-point combinator to the source signature. This complicates the specification process as the focus is no longer only on the specification of the interfaces of the data types but also on the function spaces that are manipulating them. One solution for cases when we want to keep the signatures simple and we want to enrich them with "new" functions, is to add functions as non-nullary operations even if we have

an explicit type hierarchy. The use of the implicit meta type hierarchy will not force us to interpret the newly added symbols directly in the carriers of the original models but rather as functions on those carriers. Doing that will give us greater flexibility and we will be able to express not only functions that are term definable from the source signature, but all functions that do not break abstractions imposed by that signature. Global implementations of functions as constants require that these are strictly generated from the operations of the source signature. Global implementations of functions as non-nullary operations require only that what can be observed from them should be definable.

We will see next an example of a construction which cannot be written as an implementation of a constant symbol. In other words there is no global construction between STATE and STATEITER below. Then we will see that the implementation of an iterator function is possible if we choose to represent that function as an uncurried operation or if we add a fix-point combinator to the source signature.

hspec STATE
  sorts $State, Elem$
  operations $init : State$
    $next : State \to State$
    $obs : State \to Elem$
    $case : Elem \Rightarrow (State \Rightarrow Elem) \Rightarrow (State \Rightarrow Elem)$
  constructors $init, next$
  observers obs
  combinatorial
  axioms $init \neq next(x)$
    $next(x) = next(y) \longrightarrow x = y$
    $case(e)(f)(init) = e$
    $case(e)(f)(next(x)) = f(x)$
end

spec STATEITER = STATE **then**
  operations $iter : (Elem \Rightarrow Elem) \Rightarrow Elem \Rightarrow State \Rightarrow Elem$
  axioms $iter(n)(i)(init) = i$
    $iter(n)(i)(next(x)) = n(iter(n)(i)(x))$
end

There is no global construction k : STATE⇒STATEITER because such a construction will require that the interpretation of *iter* is definable as a lambda term formed only with the *init*, *next* and *obs* operations from STATE. Informally an iterative function $iter(n)(i) : State \Rightarrow Elem$ could be defined by a different lambda term for each argument, and no single finite term can define it for all arguments. More formally, by finding a correspondence for each construction k : STATE⇒STATEITER which is not preserved be $k$we will prove that no such construction is global.

Proposition 5.9. *Let k : STATE⇒STATEITER be a tight construction. Then k is not stable.*

Proof. Assume that k is stable and let $AI \in$ STATE be the model that interprets the sorts $State$ and $Elem$ as the set of natural numbers, the higher order sorts as the corresponding full function spaces, *next* as the successor function and *obs* as the identity on the natural numbers. We define a quasi-vertical correspondence ρ: $AI \leftrightarrow AI$ by letting it be identity on basic sorts $State$ and $Elem$; on $State \Rightarrow State$ it relates only those functions that grow linearly, i.e. $f \ \rho \ f$ if and only if $f(x) - f(y) \leq 2(x-y)$ for all $x, y \in \mathbb{N}$; on other higher order sorts it is defined logically keeping in mind the previous defined restrictions. It is easy to check that ρ is a quasi-vertical correspondence.

Now let f be the doubling function, i.e. $f = \lambda x.2x$. It is clear that $f \ \rho \ f$. From the stability of k we get that $A_{iter}(f)(1) \ \rho \ A_{iter}(f)(1)$. But from the axioms we have that $A_{iter}(f)(1)(n) = f^n(1)$ for all $n \in \mathbb{N}$. Hence, $A_{iter}(f)(1) = 2^x$ and furthermore $2^x \ \rho \ 2^x$ which contradicts the definition of ρ because 2^x grows exponentially. □

In order to implement an iterator as a higher order constant we need some stronger assumptions in the source signature, for example an iterator can be implemented with the aid of a fix-point combinator. For that consider a construction k : STATEY⇒STATEYITER with:

spec STATEY = STATE **then**
  operations $Y : (TypeIter \Rightarrow TypeIter) \Rightarrow TypeIter$
  axioms $F(Y(F)) = Y(F)$
end

and

spec STATEYITER = STATEY **then**
  operations $iter : (Elem \Rightarrow Elem) \Rightarrow Elem \Rightarrow State \Rightarrow Elem$
  axioms $iter(n)(i)(init) = i$
    $iter(n)(i)(next(x)) = n(iter(n)(i)(x))$
end

where $TypeIter$ denotes the type of the iterator, i.e. $(Elem \Rightarrow Elem) \Rightarrow Elem \Rightarrow State \Rightarrow Elem$.

Please notice that we can define the iterator function with the aid of the fix-point combinator and using the formula

$$iter = Y(\lambda it.\lambda f.\lambda e.\lambda s.case(e)(\lambda x.f(it(f)(e)(x)))(s))$$

.

As we have already anticipated, an alternative to enriching the source signature is to modify the target signature and present the iterator as an uncurried function. With this modification in place we will be able to define a construction k : STATE$\Rightarrow$STATEITER'. That is because definability in the case of operations allows different defining terms depending on the arguments.

spec STATEITER' = STATE **then**
  operations $iter : Elem \Rightarrow Elem, Elem, State \rightarrow Elem$
  axioms $iter(n,i,init) = i$
    $iter(n,i,next(x)) = n(iter(n,i,x))$
end

We can implement $iter$ as an uncurried operation by letting $iter(n)(i)(x)$ be equal to $n^m(i)$ where m is a natural number such that $x = next^m(init)$ (the existence of such a number is guaranteed because *next* and *init* are the constructors of sort $State$).

Finally, let us say that a simpler signature from which to start the construction is not just an aesthetic desideratum but also has a pragmatic value. For simpler signatures it is easier to show that a relation is a correspondence, i.e. it is preserved by all operations from the signature.

We can think how tedious the proof of algebraicity would be in a signature with

a fix-point operator. Typically, combinators have complicated types and hence the relation must be declared at least for all the subtypes that appear in the combinator's type. Moreover, they are introduced for an infinity of types in a generic way for all types of the type hierarchy. In the case of the fix-point operator a definition would be $Y : (s \Rightarrow s) \Rightarrow s$ for all $s \in Types(S)$. For defining a relation for all these types a recursive definition based on the type structure is needed in order to deal with the infinity of types. Logical relations are recursively defined on the structure of types and the definition for base types is sufficient; for the higher order types the definition follows through the extensionality principle. While logical relations work well with $\mathcal{S}, \mathcal{K}$-combinators, as these combinators preserve them trivially, they do not simplify the task when used in conjunction with the fix-point combinator. It is not automatic that fix-point combinators preserve logical relations. Several studies [Wad89, Pit00, CH07] have been carried out about specializations of logical relations that work well with fix-point combinators, but they all require a more concrete framework for talking about fix-points, namely complete partial orders. In those frameworks one works with pointed relations, that relate the least elements of the cpo's, and also continuous relations, that relate the upper bounds of related chains. For us, that means that we should include in all our specifications the specification of a cpo (which requires operations of infinite arity for the upper bound) and then discuss algebraicity of our relations in these enriched signatures. Since that is a major complication we prefer as a methodological guideline to keep our signatures simple and to look for implementations of functions represented as uncurried operations, rather than including fix-point combinators in the signature and to look for implementations of functions as higher order constants.

5.4 Logical implementation of functions as higher order constants

In the previous section we have seen how the theory developed in Chapter 4 interacts with higher order types. Basically we applied a first order treatment to higher order concepts, but in this section we will look at a proper setting for reusable constructions based on functional signatures. The first step for doing that is to work with logical relations. Those are the kind of relations that are used in describing parametricity for System F and the original "theorems for free". The goal is to define a kind of global construction that will be characterized by preservation of logical relations.

In order to use logical relations to characterize a kind of global construction we should consider the main property that distinguishes them from plain algebraic relations, i.e. extensionality. Therefore we will look at global constructions that can be used only in extensional contexts. We will prove that logical correspondences (defined below as some kind of logical relation) characterize reusability of constructions in extensional contexts. Furthermore, in Section 5.5, we will prove that logical correspondences provide a sound and complete method for proving that two algebras are observational equivalent. In order to obtain such a result it is essential to work with correspondences that are first of all vertical correspondences as the existence of a vertical correspondence between to algebras guarantees that they are **COL** isomorphic (see Proposition 4.15). Hence, our setting for the rest of the chapter will be that of constructions reusable in vertically fitted extensional contexts. We will call that kind of context a *logical context* and we will use this terminology whenever it feels more natural. For example we will use *logical signature* for a functional **COL** signature, *logical signature morphism* for a vertical signature morphism between logical signatures, and *logical lifting pushout* for a fitting pushout along logical signature morphisms:

$$\begin{array}{ccc} \Sigma I'_{\mathbf{COL}} & \xrightarrow{\sigma'_{\mathbf{COL}}} & \Sigma'_{\mathbf{COL}} \\ \uparrow{\scriptstyle \phi} & & \uparrow{\scriptstyle \phi'} \\ \Sigma I_{\mathbf{COL}} & \xrightarrow[\sigma_{\mathbf{COL}}]{} & \Sigma_{\mathbf{COL}} \end{array}$$

Definition 5.10 (Logical signature morphism)**.** *Let* $\Sigma_{\mathbf{COL}}$ *be a functional* **COL** *signature. We can also say that* $\Sigma_{\mathbf{COL}}$ *is a* logical signature*. A* logical signature morphism *is a vertical signature morphism between two logical signature.*

We define *logical global constructions* as those constructions that are usable across all logical lifting pushouts.

Definition 5.11 (Logical global construction)**.** *Let* $k : SPI_{\mathbf{COL}} \Rightarrow SP_{\mathbf{COL}}$ *be a persistent* **COL***-construction. We say that k is a* logical global **COL**-construction *if for every logical lifting pushout,* $\phi(k)$ *is a* **COL***-construction between* $\phi(SPI_{\mathbf{COL}})$ *and* $\phi'(SP_{\mathbf{COL}})$*.*

We can now adapt the notion of correspondence to reflect the restrictions present in the usage contexts. In order to ensure compatibility with global lifting the correspondences must incorporate the extensionality principle, and hence they should be extensional. However, there is a little twist to the definition of logical relations (see Definition 2.9) that needs to be done in order to ensure globality. Namely, logical

correspondences are extensional relations, but extensionality is required only for the elements contained in the relation.

Definition 5.12 (Domain-restricted logical relation)**.** *Let $\Sigma = (S, OP)$ be a simply typed signature. An algebraic relation $\rho: A \leftrightarrow B$ is* logical *if for all $s_0 \Rightarrow s_1 \in S$, $f \in dom(\rho)_{A,s_0\Rightarrow s_1}$ and $g \in dom(\rho)_{B,s_0\Rightarrow s_1}$ that are extensionally equal w.r.t ρ, i.e. $f(a)\ \rho_{s_1}\ g(b)$ for all $a\ \rho_{s_0}\ b$, we have that $f\ \rho_{s_0\Rightarrow s_1}\ g$.*

Basically a domain-restricted logical relation ρ between A and B is a logical relation between the subalgebras $A^{dom(\rho)}$ and $B^{dom(\rho)}$ determined by $dom(\rho)$. These kind of relation is capable of characterizing the logical global constructions.

Definition 5.13 (Logical correspondence)**.** *Let $\Sigma_{\mathbf{COL}}$ be a* **COL** *logical signature and let A, B be two $\Sigma_{\mathbf{COL}}$-algebras. A* logical correspondence ρ *between A and B is a vertical correspondence between A and B which in addition is a domain-restricted logical relation.*

Definition 5.14 (Logically stable **COL**-construction)**.** *Let $k : SPI_{\mathbf{COL}} \Rightarrow SP_{\mathbf{COL}}$ be a persistent* **COL***-construction. We say that k is* logically (closed) stable *if it extends all (closed) logical correspondences.*

The choice made for the notion of logical correspondence can be now justified formally by proving that preservation of such relations is necessary and sufficient for a logical global construction.

Theorem 5.15 (Globality is equivalent to closed-stability)**.** *Let $k : SPI_{\mathbf{COL}} \Rightarrow SP_{\mathbf{COL}}$ be a tight* **COL***-construction. Then k is logically global if and only it is logically closed-stable.*

Proof. The proof follows the essential lines that led to Theorem 4.17.

The sufficiency part, i.e. stability implies globality, is due to the fact that isomorphisms between algebras in logical contexts are closed domain-restricted logical relations (they are extensional on domain formed by the generated subalgebras) and therefore they are preserved by closed-stable constructions.

The necessity part, i.e. globality implies stability, is based on building a logical context for each logical correspondence, in which the correspondence will be extended to an isomorphism. If we follow the proof of Propositions 4.7 we will see that the only difference when building this global context is that we should not add new observers (of the form $?^b : s \to Bool$) on the higher order sorts. We do that in order to produce

a functional signature, i.e. one that has only application operations as observers for higher order sorts. Fortunately, there is no need for those additional observers as their role is performed by the extensionality of the logical relation. More explicitly, those observers were used to coerce the observational equality in the global context to be equal to the original correspondence. However, two logical relations are equal if they are equal on base types. So, it is sufficient to coerce the observational equality in the global context to be equal to the original correspondence by using additional observers only on base types, and extensionality ensures the rest. □

In conclusion, globality amounts to preservation of closed logical correspondences. Following the pattern established by Chapter 4 we can ask ourselves if we can prove more than that, in particular if non-closed correspondences are preserved as well. We don't yet have a clear answer to that question, but we will see in the next subsection that the decomposition pattern that was used in Proposition 4.15 is no longer adequate when we refer to logical correspondences.

Before discussing properties that used to work in the standard case but no longer work in the logical case let us prove a proposition that illustrates one advantage of working with logical global constructions. The fact that a logical global construction will be used just in extensional contexts offers an advantage to the implementer when writing the definition of the construction. One can be flexible in adopting an interpretation for a functional constant by choosing between extensionally equal values. To illustrate this we will prove that we can change the interpretation given by a construction to a functional constant, with an extensionally equal one, and obtain a construction that is indistinguishable from the first.

Proposition 5.16. *Let $k : SPI_{\mathbf{COL}} \Rightarrow SP_{\mathbf{COL}}$ be a logical global* **COL***-construction, let $op : s_0 \Rightarrow s_1$ be an implemented functional constant, let $AI \in SPI_{\mathbf{COL}}$, let $A = k(AI)$ and let $f = A_{op}$. Then for every $g \in A_{s_0 \Rightarrow s_1}$ such that $f(x) = g(x)$ for all $x \in A_{s_0}$, we can define a logical global construction $k' : SPI_{\mathbf{COL}} \Rightarrow SP_{\mathbf{COL}}$ such that k' only differs from k on AI and $k'(AI)_{op} = g$. Moreover, $k(AI) \equiv_{\Sigma_{\mathbf{COL}}} k'(AI)$.*

Proof. For any logical correspondence ρ: $AI \leftrightarrow BI$ we have to show that $k'(AI)_{op}\ \rho\ k'(BI)_{op}$, i.e. $g\ \rho\ B_{op}$ where $k'(BI) = B$.

Let ρ: $AI \leftrightarrow BI$ be a logical correspondence. By the logical stability of k we get that $f\ \rho\ B_{op}$. That means that for every $a\ \rho\ b$ we get that $f(a)\ \rho\ B_{op}(b)$, which from the fact that f is extensionally equal to g leads us to $g(a)\ \rho\ B_{op}(b)$. The extensionality of ρ finally gives us that $g\ \rho\ B_{op}$.

It is easy to check that the model obtained by changing the interpretation of a symbol to an extensionally equal element yields an isomorphic algebra.

□

5.4.1 Lost properties in the logical case

In this section we will look at things that do not fit the developments from the previous chapters. First, we will see that we cannot reproduce the result of Theorem 4.17, i.e. we cannot prove that logical global constructions must preserve all logical correspondences. So, in general our best result will be the preservation of closed correspondences, but in some particular cases plain stability is still obtainable. Secondly, we have to acknowledge the failure of definability results implied by preservation of logical correspondences. This is an expected result as something similar has been known in the lambda calculus literature starting from Plotkin's paper on lambda definability [Plo80].

5.4.1.1 Failure of stability

Recall how we managed to derive that global constructions must preserve all correspondences and not just the closed ones (see Theorem 4.17). We have shown in Proposition 4.15 that each correspondence $\rho\colon A \leftrightarrow B$ can be written as the composition of two closed correspondences. In the logical case we have to show additionally that the decomposition yields logical relations whenever we start from a logical relation, but that does not always happen.

In order to have an impression on how extensionality interacts with closedness we will show that the closedness property is entirely determined by the nature of the relation on the base types.

Proposition 5.17. *Let $\Sigma_{\mathbf{COL}} = (S, OP, OP_{Cons}, OP_{Obs})$ be a* **COL** *functional signature, let A, B be two $\Sigma_{\mathbf{COL}}$-algebras and $\rho\colon A \leftrightarrow B$ be a logical correspondence between them. If ρ_s is a closed relation for all base types s then ρ is a closed relation.*

Proof. Consider a higher order type $s \Rightarrow s'$, $f_0, f_1 \in A_{s\Rightarrow s'}$ and $g_0, g_1 \in B_{s\Rightarrow s'}$ such that $f_0\ \rho\ g_0$, $f_0\ \rho\ g_1$, $f_1\ \rho\ g_0$. We need to show that $f_1\ \rho\ g_1$. For that we use the extensionality property for ρ and show that for any $a\ \rho_s\ b$ we have that $f_1(a)\ \rho_{s'}\ g_1(b)$. That is a consequence of closedness of relation $\rho_{s'}$ for the following related pairs: $f_0(a)\ \rho_{s'}\ g_0(b)$, $f_0(a)\ \rho_{s'}\ g_1(b)$,$f_1(a)\ \rho_{s'}\ g_0(b)$. □

In Proposition 4.15 the closed correspondences that decompose a correspondence $\rho: A \leftrightarrow B$ were defined as $a\ \rho_A\ (a,b)$ and $(a,b)\ \rho_B\ b$ if and only if $a\ \rho\ b$. The problem is that ρ_A and ρ_B are not necessarily logical even if ρ is logical. So, that path of reasoning does not ensure that every logical correspondence can be decomposed into two closed logical correspondences.

An alternative decomposition strategy would be to first decompose the relation ρ into closed relations only on the base types and then let the definition on the higher order types follow logically. These extensions will be closed in conformance with Proposition 5.17, but their composition will typically be bigger than ρ and hence they do not offer a sound decomposition. This phenomenon is related to the discussion in [HLST00] about the fact that logical relations are not closed under composition.

We have explained why the proof that lead to the necessity of preservation for all correspondences cannot easily be adapted for the logical case. However, that does not constitute a definite argument for showing that global logical constructions do not preserve all correspondences. We should come up with a counterexample where such a construction does not preserve a non-closed logical relation to confirm our hypothesis, but until now the search for such a counter example remains inconclusive. On the other hand we can make the decomposition work for a special class of algebras which we will call *state-injective algebras*, as described in the following paragraphs.

Definition 5.18 (State-injective algebra)**.** *Let* $\Sigma_{\mathbf{COL}} = (S, OP, OP_{Cons}, OP_{Obs})$ *be a* **COL***-signature and let A be a* $\Sigma_{\mathbf{COL}}$*-algebra. We say that A is an* state-injective algebra *if for all* $op : w \to s \in OP$ *with* $s \in S_{State}$ *we have that* A_{op} *is an injective function, i.e.* $A_{op}(\underline{a}) = A_{op}(\underline{b})$ *implies* $\underline{a} = \underline{b}$.

With additional assumptions we can use the same style of decomposing an arbitrary correspondence into two closed correspondences as in the proof of Proposition 4.15, and the correspondences we end up with will be logical. In order to obtain a sound decomposition we will be forbid the presence of types of the form $s \Rightarrow v$, where $v \in S_{Obs}$, in the signature.

Definition 5.19 (Signature with no functional observers)**.** *Let* $\Sigma_{\mathbf{COL}} = (S, OP, OP_{Cons}, OP_{Obs})$ *be a functional* **COL***-signature. We say that* $\Sigma_{\mathbf{COL}}$ *is a* signature with no functional observers *if there is no type* $s \Rightarrow v \in S$ *with* $v \in S_{Obs}$.

In signatures with no functional observers we can show that logical correspondences between state-injective algebras can be decomposed into two closed correspondences. This result might seem over restrictive and not very useful, but state-injective

algebras are quite easy to obtain and we will be able to extend the result also for standard algebras.

Proposition 5.20 (Decomposition of logical correspondences)**.** *Let* $\Sigma_{\mathbf{COL}}$ *be a* **COL** *signature with no functional observers, let* A, B *be two state-injective* $\Sigma_{\mathbf{COL}}$*-algebras and* $\rho: A \leftrightarrow B$ *be a logical correspondence between them. Consider* T_ρ*, such that* $A \equiv_{\Sigma_{\mathbf{COL}}} T_\rho \equiv_{\Sigma_{\mathbf{COL}}} B$*, and the two closed correspondences* $\rho_A: A \leftrightarrow T_\rho$ *and* $\rho_B: T_\rho \leftrightarrow B$*, such that* $\rho_A;\rho_B=\rho$*, that exist in conformance with Proposition 4.15. Then* ρ_A *and* ρ_B *are logical.*

Proof. We will only show that ρ_A is logical as the proof for ρ_B is similar.

Please recall from Proposition 4.15 that $\rho_A: A \leftrightarrow T_\rho$ is defined as $a\ \rho_A\ (a,b)$ if $a\ \rho\ b$ and consider a higher order type $s_0 \Rightarrow s_1 \in S$. Because $\Sigma_{\mathbf{COL}}$ is a signature with no functional observers we get that $s_1 \in S_{State}$. Therefore, the application that corresponds to $s_0 \Rightarrow s_1$ should be an injective function in A. Now, consider $f \in A_{s_0 \Rightarrow s_1}$ and $(g,h) \in (T_\rho)_{s_0 \Rightarrow s_1}$ such that for all $a\ \rho_A\ (a,b)$ we have that $f(a)\ \rho_A\ (g(a),h(b))$. From the definition of ρ_A we get that $f(a) = g(a)$ which by injectivity of the application means that $f = g$ and, finally from the fact that $(f,h) = (g,h)$ and $f\ \rho_A\ (f,h)$ we get $f\ \rho_A\ (g,h)$. □

The nice thing about state-injective algebras is that they are not that hard to find and hence the previous proposition has a broad applicability. In fact for any **COL** algebra we can find an isomorphic state-injective algebra.

Proposition 5.21 (Isomorphic state-injective algebra)**.** *Let* $\Sigma_{\mathbf{COL}} = (S, OP, OP_{Cons}, OP_{Obs})$ *be a* **COL***-signature and let A be a* $\Sigma_{\mathbf{COL}}$*-algebra. There exists a* $\Sigma_{\mathbf{COL}}$ *state-injective algebra* T_A *such that* $A \equiv_{\Sigma_{\mathbf{COL}}} T_A$.

Proof. We will define T_A almost like the term algebra over elements from A. However, we must alter the definition for visible sorts as visible elements must correspond one to one in order to ensure that $A \equiv_{\Sigma_{\mathbf{COL}}} T_A$. We will write T instead of T_A in the definition for simplicity of notation.

on sorts $s \in S$

$s \in S_{State}\ \ T_s = T_\Sigma(A)_s$

$s \in S_{Obs}\ \ T_s = A_s$

on operations $op : s_1, \ldots, s_n \to s \in OP$

$$s \in S_{State} \;\; T_{op}(t_1,\ldots,t_n) = op(t_1,\ldots,t_n)$$

$$s \in S_{Obs} \;\; T_{op}(t_1,\ldots,t_n) = A_{op(t_1,\ldots,t_n)}$$

Because T_A is like the term algebra on hidden sorts we get that T_A is state-injective. Moreover, T_A is isomorphic to A as it does not introduce or eliminate from its "behaviors". □

Theorem 5.22 (Globality implies stability)**.** *Let $k : SPI_{\mathbf{COL}} \Rightarrow SP_{\mathbf{COL}}$ be a tight* **COL***-construction, such that $\Sigma I_{\mathbf{COL}}$ is a functional signature with no functional observers and $SPI_{\mathbf{COL}}$ is iso-closed. Then k is a logically stable construction if it is a logical global construction.*

Proof. We use the fact that any vertical logical correspondence $\rho: A \leftrightarrow B$ can be decomposed via four closed logical correspondences via the following path: $A \leftrightarrow T_A \leftrightarrow T_\rho \leftrightarrow T_B \leftrightarrow B$ (first we use the induced relation between the state-injective algebras $T_A \leftrightarrow T_B$ given by Proposition 5.21, and then we decompose it according to Proposition 5.20). □

5.4.1.2 Failure of definability

Another result that fails to work in the logical case is definability. Again we can have a glimpse of why the failure happens if we look at the proofs for definability in the non-extensional cases. In order to derive definability results we used some minimal, i.e. term generated, correspondences and we have used the fact that implemented operations should be in these generated correspondences. The problem is that, typically, generated relations are not logical, and hence they need not be preserved by logical constructions. Again we can say that this is just an informal argument for why such a proof fails in the logical case. Fortunately we have a counterexample, of a logical global construction that is not definable.

hspec EXAMPLE = BOOL **then**

operations $u : s$

$v : s \to (s \Rightarrow Bool)$

$w : (s \Rightarrow Bool) \to s$

observers v

combinatorial

axioms $\exists x : s.x \neq u$

$\forall x, y.x \neq u \wedge y \neq u \longrightarrow x = y$

$\forall x, y.x = y \longrightarrow v(x)(y) = true$

$\forall x, y.x \neq y \longrightarrow v(x)(y) = false$

end

spec EXAMPLEEXT = EXAMPLE **then**

operations $r : s \Rightarrow s$

end

We define the construction k : EXAMPLE⇒EXAMPLEEXT by taking for each source algebra A the interpretation $A_r = \lambda x.w(v(x_A))$ where x_A is the element required by the axioms to exist and to be different from A_u.

Proof. We will prove that k is logically stable. For that consider a logical correspondence ρ: $A \leftrightarrow B$ and we will prove that k extends ρ.

We will first show that $A_v(x_A)\ \rho\ B_v(x_B)$. Notice that our higher order sorts are loose and that means that the domain of ρ on higher order sorts is the full carrier. Hence, because ρ is a domain-restricted logical relation it is sufficient to prove that $A_v(x_A)(a)\ \rho\ B_v(x_B)(b)$ for all $a\ \rho\ b$.

Notice that due to the axioms of the specification, in algebra A the elements of sort s are either observationally equal to x_A or to A_u. It is easy to see that only elements from similar groups can be related by a correspondence, i.e. $a\ \rho\ b$ implies either that $a \approx A_u$ and $b \approx B_u$ or it implies that $a \approx x_A$ and $b \approx x_B$. Therefore we will analyze the two cases. Firstly, assume that we have $a\ \rho\ b$ and $a \approx A_u$ and $b \approx B_u$. This means that $A_v(x_A)(a) = false$ and $B_v(x_B)(b) = false$ and hence $A_v(x_A)(a) = B_v(x_B)(b)$. Secondly, assume that we have $a\ \rho\ b$ and $a \approx x_A$ and $b \approx x_B$. This means that $A_v(x_A)(a) = A_v(x_A)(x_A) = true$ and $B_v(x_B)(b) = B_v(x_B)(x_B) = true$ and hence

$A_v(x_A)(a) = B_v(x_B)(b)$. Using the fact that ρ is logical from $A_v(x_A)(a)\ \rho\ B_v(x_B)(b)$ for all $a\ \rho\ b$ we get that $A_v(x_A)\ \rho\ B_v(x_B)$.

Because $A_v(x_A)\ \rho\ B_v(x_B)$ we get that $A_w(A_v(x_A))\ \rho\ B_w(B_v(x_B))$ and furthermore, from the definition of r, we get $A_r\ \rho\ B_r$. Hence, we can conclude that the construction is logically stable.

However, suppose we take a particular model for this specification by letting $A_s = \{a_0, a_1\}$, $A_u = a_1$, $A_v = eq_{A_s}$ and $A_w(eq_{a_0}) = a_0$ and $A_w(x) = a_1$ otherwise. For the model A the element x_A, different from A_u, is a_0. We can easily see that the implemented function $A_r = A_w(A_v(a_0))$ is behaviorally equal to $\lambda x.a_0$. But, there is no lambda term formed only from u, v, w that can produce a value that is behaviorally equal to $\lambda x.a_0$. □

The previous counterexample relies heavily on the fact that the appropriate logical correspondences are bi-surjective on higher order sorts. That is ensured by the fact that the higher order sorts are loose. The question of definability remains opened if we restrict to specifications for which the higher-order sorts are constructed. I seems plausible that a definability result (like visible definability Definition 4.22) could be obtained for logical constructions with constructed higher order sorts, even if they do not preserve all vertical correspondences and only preserve the ones the are logical on their domain. That would not contradict the fact that logical relations do not ensure definability, because our special notion of domain-restricted logical relations are more adapted to enforce definable elements. However, this work remains to be developed in the future.

5.5 Representation independence

In this section we consider the issue of providing a sound and complete method for proving observational equivalence of extensional models. This problem has been studied both by the lambda calculus and by the algebraic specification communities and we will briefly recap the history of the problem.

The behavior of programs represented as lambda terms has been studied mainly for structures which satisfy the extensionality axiom literally, i.e. the Henkin models typically used to give semantics to lambda calculus. An important technique used for proving the fact that the behavior of programs does not depend on the way data is represented was developed by Mitchell in [Mit91]. Mitchell's theorem of representation

independence states that two algebras over a signature, that does not contain higher order functions, are observational equivalent if and only if there exists a logical relation between them which is a bijection on visible sorts. Furthermore, an example was presented in [Mit96] to show that the requirement for the absence of higher order operations cannot be eliminated.

To summarize the known facts on this topic we can say that the existence of a logical relation, which is also a bijection on visible sorts, is a sufficient condition for showing observational equivalence between algebras. It is also a necessary condition if there are no higher order functions, and there are simple counterexamples which show that the restriction to first order functions cannot be discarded (Section 8 of [HS02] also contains a succinct recapitulation of these results). Further study tried to change the notion of relation in order to obtain a complete method for proving observational equivalence between extensional models. Honsell and Sannella [HS02] proposed for this a lax definition based on logical relations: the *prelogical relations*. Another approach was devised by Hannay in his PhD thesis [Han01], where he proposed *abstraction barrier observing relations* as a complete method for showing observational equivalence between algebras represented as packages of an existential type in System F.

Before presenting our approach we will illustrate the problem by presenting two observationally equivalent models for which there is no logical relation between them that is a bijection on visible sorts.

Example 5.23 (No logical relation)**.** *Let* $\Sigma_{\mathbf{COL}}$ *be the following* **COL** *signature:*

spec NOLOG = BOOL ***then***
  sorts $s, s \Rightarrow s$
  operations $c : s$
    $op_0, op_1 : s \Rightarrow s$
    $obs : s \Rightarrow s \rightarrow Bool$
    $app : s \Rightarrow s, s \rightarrow s$
  observers $obs, (app, 1)$
  constructors c
end

let A be the following $\Sigma_{\mathbf{COL}}$*-algebra:*

$$A_s = \{a, b\},\ A_{s \Rightarrow s} = \{f, g\}$$

$$A_c = a,\ A_{op_0} = f,\ A_{op_1} = g,\ A_{obs} = [f \mapsto true, g \mapsto false]$$

and let B be the following Σ*-algebra:*

$$B_s = \{a,b\},\ B_{s\Rightarrow s} = \{f,g\}$$

$$B_c = a,\ B_{op_0} = g,\ B_{op_1} = f,\ B_{obs} = [f \mapsto \mathit{false}, g \mapsto \mathit{true}]$$

with:

- $A_{app}(f,x) = B_{app}(f,x) = a$ *for all* $x \in \{a,b\}$
- $A_{app}(g,x) = B_{app}(g,x) = b$ *for all* $x \in \{a,b\}$

We can show that A and B are observational equivalent, i.e. $A \equiv_{\Sigma_{\mathbf{COL}}} B$*, but there is no logical relation that is identity on Bool between them. Moreover, it is clear that A and B satisfy the extensionality axiom literally.*

Proof. Assume there exists a logical relation ρ: $A \leftrightarrow B$. Because $A_c\ \rho\ B_c$ we have that $a\ \rho\ a$ which by the extensionality of ρ leads to $f\ \rho\ f$. Based on this we get that $A_{obs}(f)\ \rho\ B_{obs}(f)$, which is a contradiction because it is equivalent to *true* ρ *false*. □

Please have in mind that in the previous example, $\Sigma_{\mathbf{COL}}$ is not a combinatorial signature and hence we cannot construct terms corresponding to arbitrary lambda terms. That is important, because otherwise we will have to declare A and B as having different behaviors due to $A_{obs(\lambda x.c)} \neq A_{obs(\lambda x.c)}$, but since $\lambda x.c$ is not a term of our signature we do not need to care about that mismatch. Examples for full lambda-calculus can be found in [Mit96, HS02].

The previous example illustrates the problem that was investigated in the literature, i.e. two observationally equivalent extensional models that cannot be related logically. While this is a valid statement we claim that the goal was not chosen with care, and a solution to this problem arises naturally if we ask the right question.

The deficiency of the previous counterexample is that it tries to find a logical relation between algebras that are literally extensional rather than behaviorally extensional. That was not noticed before, because the frameworks that where used to present the issue do not use the concept of observational equality between elements as a primary concept. The extensionality axiom is regarded as belonging to the meta-framework of lambda-calculus, and therefore literal extensional models (Henkin models) are used for the semantics of lambda-calculus. We want to underline that extensionality is an axiom that *should* be adapted to the kind of semantics that is in use. Hence, when one wants to study observational equivalence one should use behavioral extensionality. We

will see that if we only consider behavioral extensional models then logical relations are necessary for observational equivalence.

Also, the notion of logical relation should be aware of the fact that not all the elements in the carriers of algebras are of interest. Basically, one has to work with domain-restricted relations in order to guarantee completeness.

We will show that in logical signatures two **COL**-algebras are isomorphic if and only if there exists a logical correspondence between them.

Proposition 5.24 (Soundness of logical correspondences)**.** *Let* $\Sigma_{\mathbf{COL}}$ *be a logical* **COL***-signature and let A and B be two* $\Sigma_{\mathbf{COL}}$*-algebras. If there exists a logical correspondence* $\rho: A \leftrightarrow B$ *then A and B are* **COL***-isomorphic*

Proof. Assume that there exists a logical correspondence $\rho: A \leftrightarrow B$. By definition this means that ρ is a vertical correspondence and a consequence of Proposition 4.15 is that vertical correspondences enforce **COL**-isomorphism. □

Now we can prove that the existence of a logical correspondence is a necessary condition for observational equivalence. The only requirement is to work in a logical **COL** signature, which basically means to require behaviorally extensional algebras.

Proposition 5.25 (Completeness of logical correspondences)**.** *Let* $\Sigma_{\mathbf{COL}}$ *be a logical* **COL***-signature and let A and B be two isomorphic* $\Sigma_{\mathbf{COL}}$*-algebras. Then the isomorphism* $h: A \to B$ *is a logical correspondence.*

Proof. In order to show that h is a domain-restricted logical relation we have to show that all functions from the generated algebras that are extensionally equal w.r.t. h are also related by h.

Let $f \in \langle Gen_{A_{\mathbf{COL}}}(\Sigma)\rangle_{s_0 \Rightarrow s_1}$ and $g \in \langle Gen_{\Sigma_{\mathbf{COL}}}(B)\rangle_{s_0 \Rightarrow s_1}$ such that whenever $a\ h_{s_0}\ b$ we have $f(a)\ h_{s_1}\ g(b)$. Using the fact that h is an isomorphism, there exists $f' \in \langle Gen_{\Sigma_{\mathbf{COL}}}(B)\rangle_{s_0 \Rightarrow s_1}$ such that $f\ h_{s_0 \Rightarrow s_1}\ f'$. We will show that g and f' are observational equal.

For any $b \in \langle Gen_{\Sigma_{\mathbf{COL}}}(B)\rangle_{s_0}$ let a be an element in $\langle Gen_{\Sigma_{\mathbf{COL}}}(A)\rangle_{s_0}$ such that $a\ h_{s_0}\ b$. For this pair we get that $f(a)\ h_{s_1}\ g(b)$ from the assumptions for f and g, and $f(a)\ h_{s_1}\ f'(b)$ from the congruence properties of h. Now, using the closure property of h we can deduce that $g(b) \approx_{\Sigma_{\mathbf{COL}},B} f'(b)$. Because *app* is the only observation we get that $g \approx_{\Sigma_{\mathbf{COL}},B} f'$. Finally, the properties of the **COL**-morphism ensures us that $f\ h_{s_0 \Rightarrow s_1}\ g$. □

We can gather the previous results to express the equivalence between **COL** isomorphisms and the existence of a logical correspondence.

Theorem 5.26 (**COL**-isomorphisms and logical correspondences)**.** *Let* $\Sigma_{\mathbf{COL}}$ *be a logical* **COL***-signature and let A and B be two* $\Sigma_{\mathbf{COL}}$*-algebras. Then A and B are* **COL***-isomorphic if and only if there exists a logical correspondence between them.*

It is worth pointing out that vertical correspondences have their domain formed from the generated subalgebras of th related algebras. In conclusion, in order to show that two algebras are observationally equivalent one has to define a relation on basic sorts and by restricting the domain only to generated elements the definition on higher order sorts follows by extensionality.

Logical correspondences are important because they are hierarchically defined over the structure of types, and they can be defined by only providing a definition on basic types. This compactness of definition was the reason why a characterization of observational equivalence in terms of logical relations was sought in the first place. The difference between domain-restricted logical relations and plain logical relations is that for the former one has to priorly define the domain of the relation, while for the later the domain is defined also recursively over the structure of types. However, this is not an impediment in **COL** because the domain of a logical correspondence is given by signature's constraints and do not need to be explicitly defined for every sort.

Next we will give an example of two isomorphic **COL** algebras for which there is no logical relation but we can easily define a logical correspondence. The example is taken from Jo Hannay's PhD thesis [Han01] (Example 5.7).

Example 5.27. *Let* $\Sigma_{\mathbf{COL}}$ *be the following combinatorial* **COL** *signature:*

hspec CROSSCONTAINER = BOOL + NAT ***then***
  sorts *Container*
  operations *empty* : *Container*
    insert : *Container*, *Nat* $\rightarrow$ *Container*
    isIn : *Container*, *Nat* $\rightarrow$ *Bool*
    remove : *Container*, *Nat*$\Rightarrow$*Container*
    crossover : (*Container*$\Rightarrow$*Nat*$\Rightarrow$*Container*)$\Rightarrow$*Nat*$\Rightarrow$*Bool*
  constructors *empty*, *insert*, $\mathcal{S}$, $\mathcal{K}$
  observers (*isIn*, *1*)
  combinatorial
end

let A and B be the following Σ-algebras:

- $A_{Container} = B_{Container}$ *is the set of lists on natural numbers, written as* $[7,2,3,\ldots]$.
- *On higher order types we take the full type hierarchy.*
- $A_{empty} = B_{empty}$ *is the empty list.*
- $A_{insert} = B_{insert} =$ *insertOne uniquely inserts a natural number before the first entry that is greater than it.*
- $A_{isIn} = B_{isIn} =$ *isOne is the function that checks if an number occurs only once in a list.*
- $A_{remove} =$ *delOne is the function that removes the first occurrence of a natural number from a list.*
- $B_{remove} =$ *delAll is the function that removes all occurrences of a natural number from a list.*
- $A_{crossover} = \lambda f : Container \Rightarrow Nat \Rightarrow Container.\lambda n : Nat.isIn(f([1],n))(n)$.
- $B_{crossover} = \lambda f : Container \Rightarrow Nat \Rightarrow Container.\lambda n : Nat.isIn(f([1,1],n))(n)$.

It is proved in by Hannay in [Han01] that A and B cannot be distinguished by evaluating terms but there is no logical relation between them which is identity on visible sorts. Briefly, the reason behind the failure is that a logical relation ρ between A and B will always relate finite ascending lists of natural numbers, which implies *delOne* ρ *delOne* by extensionality of ρ. Furthermore, this leads to a contradiction because $A_{crossover}(delOne,1)$ is not equal to $B_{crossover}(delOne,1)$.

The problem was solved by Hannay with the help of a new kind of relation, the *abstraction barrier-observing relations* ([Han03]). These relations are very similar to our logical correspondences (except they are defined for System F) and we will explain why there is a logical correspondence between A and B.

We will prove that even if there is no logical relation between A and B we can define a logical correspondence between them.

The difference in definitions between logical relations and logical correspondences is that the former relate any two functions that are extensionally equivalent, while the latter require this only for functions in the domain of the relation. Basically, in order to define a logical correspondence ρ one has to:

Define the domain: Define the domain of the relation ρ.

Define the basic relation: Define the related elements on basic sorts. The related elements on higher order sorts follow by extensionality.

Prove algebraicity: Prove that ρ is algebraic, i.e. that all the operations in the signature (except combinators, see explanation next) preserve the relation. The proof of algebraicity is significantly simplified by the fact that on higher order sorts the relation is defined by extensionality. It can be easily proved that combinators are always in relation (due to the axioms that define them).

For the algebras in Example 5.27 we can follow these steps:

Define the domain: Let $dom(\rho) \subseteq A \times B$ be formed by the restriction to the generated subalgebras, i.e. $dom(\rho) = \langle Gen_{\Sigma_{\mathbf{COL}}}(A)\rangle \times \langle Gen_{\Sigma_{\mathbf{COL}}}(B)\rangle$.

Define the basic relation: On visible sorts ρ is identity, while on *Container* sort is a sub-relation of identity relating all finite ascending lists of natural numbers.

Prove algebraicity: It is easy to prove that all first order operations preserve ρ. The only challenging thing to prove is that $A_{crossover}\ \rho\ B_{crossover}$. It does not really help to try to do a proof using extensionality of ρ because *crossover* is defined in B using elements (the list $[1,1]$) that is not in the domain of ρ. So, the solution is to do an induction proof on the structure of f. One can prove that the operations obtained by combining *insert* and *remove* in A and respectively B cannot be distinguished using *crossover*.

This example illustrates the general methodology of defining logical correspondences and hence for proving that two **COL** algebras are isomorphic.

Another example from literature is that given by Mitchell in [Mit96] (Exercise 8.5.6) which also appears in [HS02]. We will rewrite that example in **COL** to make the presentation consistent.

Example 5.28 (Mitchell's example)**.** *Let* $\Sigma_{\mathbf{COL}}$ *be the combinatorial-generated* **COL***-signature over the base type nat enriched with an operation op* : $(nat \Rightarrow nat) \Rightarrow nat$ *and the usual constructors* $0 : nat$ *and* $s : nat \Rightarrow nat$.

spec MITCHELL
  sorts Nat
  operations $0 : Nat$
    $s : Nat \to Nat$
    $op : (Nat \Rightarrow Nat) \Rightarrow Nat$
  combinatorial
  constructors $\mathcal{S}, \mathcal{K}, app, 0, s$
end

Let A be the full type hierarchy over $A_{Nat} = \mathbb{N}$ with A_{op} interpreted as the constant function 0, i.e. $A_{op}(f) = 0$ for all $f \in A_{Nat \Rightarrow Nat}$. Let B be like A but with $B_{op}(f) = 0$ if f is computable and $B_{op}(f) = 1$ otherwise. Since the difference between A and B cannot be detected by evaluating terms we have that $A \equiv_{\Sigma_{\mathbf{COL}}} B$. Moreover we cannot find a logical relation between the full carriers of A and B which is identity on $\mathbb{N}$: any logical relation ρ will relate equal functions from A and B, i.e. for all $f \in \mathbb{N} \Rightarrow \mathbb{N}$ we have that $f \; \rho \; f$, and that leads to 0 being in relation to 1 by applying A_{op} and B_{op}.

We can easily observe that the failure in finding a logical relation is due to the fact that we are willing to accept in the relation functions that are extensionally equivalent even if they are not in the generated part of the algebras. The definition of domain-restricted logical relations do not force use to do that. In fact, if we restrict only to the generated parts then we can find a logical relation.

5.6 Summary

In this chapter we have looked at global constructions that work in contexts with an explicit type hierarchy. Treating functions as constant symbols on higher order types brought us closer to the setting of lambda calculus.

We have discussed the notion of logical relation from the perspective of global constructions. This gives a new characterization of (closed) logical relations, as being the kind of relations that are necessary and sufficient to enable a construction to be usable in all logical contexts. The extensional nature of the logical relations is however a hurdle in the way of obtaining smooth characterization results. In particular we witness the failure of stability and definability of global constructions, but the former can be recovered for a particular kind of signatures (i.e. those with no functional observers).

In the end, we have presented a new approach to a long-standing problem concerning a sound and complete method for proving observational equivalence between extensional models. Our approach is to define a new kind of relation that is in conformance with **COL** constraints and to prove that if one looks for logical relations also on subalgebras of the equivalent algebras then the method is complete.

Chapter 6

Conclusions

Contents

This is the final chapter of this thesis. It contains a short recapitulation of what has been presented and what can still be done in order to develop the theory of constructions.

6.1 Summary

In the previous chapters we have investigated the properties of global constructions. This topic has been researched under various forms in [Sch87, BST08] but we bring a new view on the subject by casting its problems in the framework of *Constructor-based Observational Logic*.

The choice of **COL** is not just an aesthetic preference or just for the sake of a small variation, but during the thesis it can be observed that it is the optimum framework for expressing such results. **COL** allows us to be explicit about the elements in an algebra whose behavior is of interest and about the operations that are used to observe that behavior. This fine level of detail in specifying behaviors enabled us to identify some issues that were undetected in previous presentations of the subject. Recall one of the results presented in Section 5.5, in which we provide a sound and complete method based on logical relations for proving observational equivalence (**COL**-isomorphism) between algebras. That problem has been investigated in both lambda calculus and algebraic communities without satisfactory results. However, the solution is almost

trivial in **COL**, and the reason why that happens is exactly because we are using its specific features to control the behavior of the algebras involved . It is essential to our solution that the problem is expressed for algebras of *logical signatures* (see Definition 5.10) and the definition of logical signatures uses the features of **COL**, i.e. the distinguished single observer on the higher order sorts that must be application and the requirement for higher order sorts to be loose. That conjunction of requirements could not be easily expressed in previous frameworks where practically all sorts were constrained and all operations were observers (see Definition 2.34). It is the merit of the framework that it allows such a simple solution to the original problem.

The richness of **COL** signatures and of the signature morphisms between them also allowed us to examine several flavors of globality (genericity), depending on the details of the fitting morphisms. For each of these variants, globality is characterized in terms of preservation of correspondences which reflect in their structure the choices of global contexts.

In Chapter 3 we have presented the constructions that can be used in any global context. The extreme flexibility in the use of those constructions means that the implementer is highly restricted. He cannot assume anything more than what it is present in the signature and hence in order to write a general global construction he will have to restrict himself to implementing operations that are definable in terms of the source signature. This requirement greatly reduces the number of general global constructions that can be written, but on the other hand it is a powerful tool for the user of such construction in order to prove properties of constructed models. Overall, general constructions do not interact well with behavioral abstraction, e.g. we have witnessed in Example 3.15 the failure of uniform definability even for iso-closed specifications. That should not come as a surprise as it happens because we can use general global constructions in unrestricted contexts, i.e. without paying any attention to the **COL**-aspects of the source signature.

Chapter 4 is designed to address the shortcomings of general global constructions. The studied constructions take into account the behavioral details present in their contexts. For example, vertical constructions can be used only in contexts that preserve the nature of the loose and visible sorts. That requirement translates immediately into the definition of the relations characterizing vertical global constructions. We define vertical correspondences to be bi-surjective on loose sorts and bi-injective on visible sorts in order to capture the characteristics of all **COL**-isomorphisms between algebras from vertically fitted contexts. Returning to roles, from the point of view of the imple-

menter we can remark that the definability result is less restrictive, i.e. one can write implementations using elements from the loose carriers (see the loose-definability result Proposition 4.10). Of course, that liberty for the implementer comes with a cost for the user of vertical global constructions who will have less knowledge to use when proving properties of constructed models. However, this drawback is not to bad as in the majority of cases, i.e. when specifications are closed under **COL**-isomorphisms, a stronger definability result (visible definability) can be inferred, and hence some power is regained for the user of global constructions. In the final part of the chapter we look at some hybrid constructions, situated between the two notions that we have already discussed, namely the quasi-vertical constructions. Quasi-vertical constructions can be used in more contexts than the vertical constructions, i.e. the only requirement is to keep the nature of visible sorts, but they take on some of the good properties of vertical constructions.

The last chapter on global constructions is Chapter 5 in which we describe the interaction between global constructions and higher order sorts. We find out that by considering extensionality of the contexts we get logical relations as the characterizing correspondences. However, for such constructions we do not get definability properties. Nevertheless, the study of higher order sorts in conjunction with the notion of **COL**-constructions led us to the concept of logical signatures and to a complete characterization of observational equivalence in terms of logical relations.

6.2 Future work

Some areas of the subject are left untouched by this thesis. Due to our interest in characterizing the definability of new operations in terms of operations already present in the signature we have restricted ourselves to constructions along signature morphisms that do not add new sorts. Extending the interest to constructions that also add new sorts should lead to definability results for types, not just for operations. Just anticipating the results that might come out of this study we can expect that the carriers of the newly added sorts would be freely generated from the ones of the old sorts. It remains to figure out what will be the conditions for the contexts that would characterize completely such type generating constructions.

6.2.1 Constructions for indexed categories

A path to follow in the further development of the theory of generic constructions would be to treat it at a categorical level. At first, the framework of institutions seems the most appropriate for talking abstractly about constructions but we would prefer a simpler setting in order to make some additional points about constructions. For that we will use the classical notion of indexed categories [Mac98]. Recall that an *indexed category* is a functor $F : \mathbb{S}yn^{op} \to \mathbb{S}em$ with $\mathbb{S}yn, \mathbb{S}em \in \mathbb{C}at$.

Starting from an indexed category F one can define the category of persistent constructions over that indexed category. The objects of the category of constructions would be the fibers of F, i.e. $F(\Sigma)$ for $\Sigma \in \mathbb{S}yn$. The morphisms between $F(\Sigma)$ and $F(\Sigma')$ are pairs $\langle \sigma, k \rangle$ such that $\sigma : \Sigma \to \Sigma'$ and $k : F(\Sigma) \Rightarrow F(\Sigma')$ such that $k; F(\sigma) = 1_{F(\Sigma)}$. The case that was studied in this thesis is that of iso-preserving constructions over the indexed category Mod that gives the semantics of **COL** specifications.

The goal of this research will be to define categorically the correspondences and hence provide a categorical characterization of global constructions in terms of preservation of correspondences. It will be interesting to find out how we can present in an indexed category F the link between the contexts of use for global constructions (represented by morphisms in $\mathbb{S}yn$) and the particular requirements on correspondences (represented by morphisms in the fibers of F), in order to explain for example the link between vertical contexts and vertical correspondences.

Another way to make the presentation of generic constructions more independent of the concrete setting and more reliant on categorical concepts will be to investigate the relation between the definability properties of those constructions and the concept of Beth definability as it appears in classical model theory, and that was previously lifted to institutional model theory in [PD06, Dia08].

6.2.2 Curry-Howard isomorphism for constructions

Another idea is to give the semantics of sentences in the same way as the semantics of signature symbols is given. That will allow us to express the Curry-Howard isomorphism at the level of institutions, that is to see sets of types (signatures) as sets of sentences. For that we will present the meaning of sentences using an indexed category, in order to mimic the way the model functor Mod is introduced for an institution [GB92], using an *evidence* functor Evd that gives all the evidence for the validity of a set of sentences. Furthermore, starting from the fact that meanings for signatures

(models) and meanings for sentences (evidence) are both given via indexed categories, we can define for both the corresponding category of persistent constructions. Hence, functions between the classes of evidence can be studied using the same theory as functions beween classes of models.

Definition 6.1 (K-institution)**.** *A* K-institution $\mathsf{KI} = (\mathbb{S}ign, \mathsf{Sen}, \mathsf{Mod}, \mathsf{Evd}, \models)$ *consists of*

- *an indexed category* $\mathsf{Mod} : \mathbb{S}ign^{op} \to \mathbb{C}at$
- *a functor* $\mathsf{Sen} : \mathbb{S}ign \to \mathbb{C}at$*, giving for each signature* Σ *a category of sentences (also written as* Sen_Σ *instead of* $\mathsf{Sen}(\Sigma)$*)*
- *for each signature* Σ*, an indexed category* $\mathsf{Evd}_\Sigma : \mathsf{Sen}_\Sigma^{op} \to \mathbb{C}at$
- *a relation* $\models_\Sigma \subseteq |\mathsf{Mod}(\Sigma)| \times |\mathsf{Sen}(\Sigma)|$ *for each* $\Sigma \in |\mathbb{S}ign|$*, called* Σ-satisfaction*, such that the satisfaction condition holds.*

The difference between a classical institution as defined in [GB92] and a K-institution consists in the fact that the sentences for each signature form a category on which is based an indexed category of evidence. This is the first step towards bringing the Curry-Howard isomorphism to the level of institutions (surely additional conditions should be satisfied by the satisfaction relation in order to ensure compatibility with the evidence functors but we will not go into that right now).

So, we define the meaning of sentences, i.e. given by Evd_Σ for each signature Σ, using the same categorical concept (indexed categories) as is used for defining the meaning of signatures, given by Mod. Hence, for each signature Σ we have objects $E, E' \in |\mathsf{Sen}(\Sigma)|$ (informally representing the (sets of) sentences over Σ) and morphisms between them (best thought as inclusions between sets of sentences). Intuitively, the objects in $\mathsf{Evd}_\Sigma(E)$ are representing evidence for the validity of the sentences in E.

Let us explain briefly the name chosen for evidence. At first we were tempted to use the term "proofs" instead of "evidence" as the title of this section shows. That was mainly motivated because the Curry-Howard isomorphism is often presented as "types as sentences" and "functions as proofs". However, we use the term "evidence" instead, because we save the term "proof" to denote a computable evidence or computable functions between evidence, as will be revealed later.

Before addressing the issue of constructions in K-institutions let us remark that we can already link some concepts from the model-theoretic world to concepts from the

proof-theoretic world in the style of the Curry-Howard correspondence. One can see the model-theoretic operation of reduction given by the action of Mod on signature morphisms, that is taking models of a larger signature into models of a smaller signature, as correspondening to the forgetful proof rule that says that evidence for a larger set of sentences can be seen as an evidence for a smaller set of sentences. In other words, the fact that E' entails E when $E \subseteq E'$ is semantically given by a reduct functor.

$$\frac{\sigma : \Sigma \to \Sigma'}{\mathsf{Mod}(\sigma) : \mathsf{Mod}(\Sigma') \Rightarrow \mathsf{Mod}(\Sigma)} \qquad \frac{\iota : E \to E'}{\mathsf{Evd}(\iota) : \mathsf{Evd}(E') \Rightarrow \mathsf{Evd}(E)}$$

As we have seen in the previous section, persistent constructions can be defined for each indexed category, so in the case of K-institutions we can talk about model-constructions, induced by Mod, but also about evidence-constructions, induced by Evd_Σ for each $\Sigma \in \mathbb{S}ign$. That is, functions between model classes correspond through the Curry-Howard isomorphism to functions between evidence classes. More speculatively, one can think of evidence-constructions as a form of theorem proving, i.e. that it is sufficient to map each evidence in $\mathsf{Evd}(E)$ to a evidence in $\mathsf{Evd}(E')$ in order to assert the validity of E' starting from that of E. Therefore, on this line, we can say that implementations correspond to proofs. The goal of applying the Curry-Howard isomorphism for K-institutions is to transfer insights from refinement theory to proof theory and vice-versa.

A theory of proofs for institutions was also proposed in [MGDT07] and then developed in [Dia06]. While that approach identifies the need to treat sets of sentences and proofs between them rather than single sentences, it does not organize sentences as an indexed category. We think that our presentation has the advantage of giving semantics to sentences in a way that embodies the Curry-Howard isomorphism naturally, but the full extent to which this view brings something useful and new still remains to be investigated.

6.2.3 Towards a notion of beliefs

Currently just at the level of speculation, we can think about interpreting into proof theory the results of global constructions obtained in refinement theory. We have already said that an evidence-construction can be seen as some kind of proof, but giving a function between $\mathsf{Evd}(E)$ and $\mathsf{Evd}(E')$ is definitely pretty far from what is typically understood by giving a proof from E to E'.

Classical proofs are typically generated from a set of proof rules, and are not just arbitrary functions. And here is the point where one can make a link with the

theory behind global constructions. More explicitly, an evidence-construction ek : $\mathsf{Evd}(E_0) \Rightarrow \mathsf{Evd}(E_1)$ can be thought as an arbitrary "proof" between E_0 and E_1 because it gives a way to see each item of evidence for E_0 as an evidence for E_1. However, we have seen that constructions that can be reused in global contexts satisfy various definability properties. One can define a notion of global evidence-construction, and obtain a definability result for such constructions. Using that insight one can imagine that global evidence-constructions correspond in fact to proofs as we understand them, i.e. adhering to a generation principle.

Our proposed hypothesis is to consider that the concept of proof can be explained using the notion of global construction. That means, first of all, that there are different notions of proofs depending on the scope of reusability. For example, the ones that can be reused in any context, having discretionary knowledge about evidence, are the universal proofs which have the property of being generated. However, one can investigate different notions of proofs, which satisfy weaker definability results. We denote these weaker variants of proofs by the term "beliefs". Beliefs are not necessary generated and can be of various kinds: vertical beliefs, quasi-vertical beliefs. For example vertical beliefs are those evidence-constructions that can be reused only across vertical lifting pushouts. These constructions have, as has been investigated in this thesis, weak definability properties (at best we have visible definability). However, in the realm of proof theory that might become interesting, because such beliefs represent an argument for proving conclusions without generating them strictly from hypotheses by using logical deduction. Beliefs might sound odd as a basis for an argumentative discourse as they are not generated by a step by step deduction. For them it is not important how they are obtained but how they can be used. And the more limited is the scope of their use, the more liberty in making the argument can be accepted. A motto for using beliefs would be: one does not need to be rigourous in one's proofs if they are not to be used in all contexts.

But as we said, all these thoughts are at the level of speculation and need to be applied to some concrete cases in order to see if there is any value in creating a notion of beliefs to complement the already well established study of proofs.

Bibliography

[Bar81] H. P. Barendregt. *The Lambda Calculus*. Netherlands: North-Holland, 1981.

[BG80] R.M. Burstall and J.A. Goguen. The semantics of Clear, a specification language. In Dines Bjørner, editor, *Proceedings of the 1979 Copenhagen Winter School on Abstract Software Specification*, volume 86 of *Lecture Notes in Computer Science*, pages 292–332. Springer, 1980.

[BH95] Michel Bidoit and Rolf Hennicker. Proving the correctness of behavioural implementations. In *Proc. AMAST '95*, volume 936 of *Lecture Notes of Computer Science*, pages 152–168. Springer, 1995.

[BH96] Michel Bidoit and Rolf Hennicker. Behavioural theories and the proof of behavioural properties. *Theoretical Computer Science*, 165(1):3–55, 1996.

[BH98] Michel Bidoit and Rolf Hennicker. Modular correctness proofs of behavioural implementations. *Acta Informatica*, 35(11):951–1005, 1998.

[BH05] Michel Bidoit and Rolf Hennicker. Externalized and internalized notions of behavioral refinement. In *ICTAC*, volume 3722 of *Lecture Notes in Computer Science*, pages 334–350. Springer, 2005.

[BH06a] Michel Bidoit and Rolf Hennicker. Constructor-based observational logic. *Journal of Logic and Algebraic Programming*, 67(1–2):3–51, 2006.

[BH06b] Michel Bidoit and Rolf Hennicker. Proving behavioral refinements of COL-specifications. In Kokichi Futatsugi, Jean-Pierre Jouannaud, and José Meseguer, editors, *Algebra, Meaning and Computation: Essays Dedicated toJoseph A. Goguen on the Occasion of His 65th Birthday*, volume

4060 of *Lecture Notes in Computer Science*, pages 333–354. Springer, 2006.

[BHW95] Michel Bidoit, Rolf Hennicker, and Martin Wirsing. Behavioural and abstractor specifications. *Science of Computer Programming*, 25(2-3):149–186, 1995.

[BST08] Michel Bidoit, Donald Sannella, and Andrzej Tarlecki. Observational interpretation of CASL specifications. *Mathematical Structures in Computer Science*, 18:325–371, 2008.

[CF58] Haskell Curry and Robert Feys. *Combinatory Logic*, volume 1. Netherlands: North-Holland, 1958.

[CH07] Karl Crary and Robert Harper. Syntactic logical relations for polymorphic and recursive types. *Electron. Notes Theor. Comput. Sci.*, 172:259–299, 2007.

[Dia06] Răzvan Diaconescu. Proof systems for institutional logic. *Journal of Logic and Computation*, 16(3):339–357, 2006.

[Dia08] Răzvan Diaconescu. *Institution-independent Model Theory*. Birkhäuser, 2008.

[EK99] Hartmut Ehrig and Hans-Jörg Kreowski. Refinement and implementation. In E. Astesiano, H.-J. Kreowski, and B. Krieg-Brückner, editors, *Algebraic Foundations of Systems Specification*, pages 201–242. Springer, 1999.

[EM85] H. Ehrig and B. Mahr. *Fundamentals of Algebraic Specification 1*, volume 6 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, Berlin, 1985.

[GB92] Joseph Goguen and Rod Burstall. Institutions: Abstract model theory for specification and programming. *Journal of the Association for Computing Machinery*, 39(1):95–146, 1992.

[Gir72] Jean-Yves Girard. *Interpretation functionelle et elimination des coupures dans l'arithmetique d'ordre superieure*. PhD thesis, Universite Paris VII, Paris, 1972.

[GLT89] Jean-Yves Girard, Yves Lafont, and Paul Taylor. *Proofs and Types*, volume 7 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1989.

[GM00] Joseph Goguen and Grant Malcolm. A hidden agenda. *Theoretical Computer Science*, 245(1):55–101, 2000.

[Gog89] J. A. Goguen. Principles of parameterized programming. *Software reusability: vol. 1, concepts and models*, pages 159–225, 1989.

[Gri90] Timothy G. Griffin. A formulae-as-type notion of control. In *POPL '90: Proceedings of the 17th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 47–58, New York, NY, USA, 1990. ACM.

[Han01] Jo Hannay. *Abstraction Barriers and Refinement in the Polymorphic Lambda Calculus*. PhD thesis, University of Edinburgh, 2001.

[Han03] Jo Hannay. Abstraction barrier-observing relational parametricity. In Martin Hofmann, editor, *Typed Lambda Calculi and Applications*, volume 2701 of *Lecture Notes in Computer Science*, pages 1086–1086. Springer Berlin, 2003.

[HB99] Rolf Hennicker and Michel Bidoit. Observational logic. In Armando Martin Haeberer, editor, *Proceedings of the 7th International Conference on Algebraic Methodology and Software Technology (AMAST'98)*, volume 1548, pages 263–277, 1999.

[HLST00] Furio Honsell, John Longley, Donald Sannella, and Andrzej Tarlecki. Constructive data refinement in typed lambda calculus. In *Proceedings of the 3rd International Conference on Foundations of Software Science and Computation Structures. European Joint Conferences on Theory and Practice of Software (ETAPS 2000)*, volume 1784 of *Lecture Notes in Computer Science*, pages 161–176. Springer, 2000.

[How80] William Howard. The formulae-as-types notion of construction. In *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 479–490. Academic Press, 1980.

[HS02] Furio Honsell and Donald Sannella. Prelogical relations. *Information and Computation*, 178(1):23–43, 2002.

[Mac98] Saunders MacLane. *Categories for the Working Mathematician*. Springer-Verlag, New York, 1998.

[Mei92] Karl Meinke. Universal algebra in higher types. *Theoretical Computer Science*, 100(2):385–417, 1992.

[Mes89] José Meseguer. General logics. In H.-D. Ebbinghaus, editor, *Logic Colloquium '87*, pages 275–329. North-Holland, 1989.

[MG94] G. Malcolm and J. Goguen. Proving correctness of refinement and implementation. *Technical Monograph PRG-114, University of Oxford*, 1994.

[MGDT07] Till Mossakowski, Joseph Goguen, Răzvan Diaconescu, and Andrzej Tarlecki. What is a logic? In Jean-Yves Beziau, editor, *Logica Universalis: Towards a General Theory of Logic*, pages 111–135. Birkhäuser, 2007.

[Mit91] John C. Mitchell. On the equivalence of data representations. *Artificial intelligence and mathematical theory of computation: papers in honor of John McCarthy*, pages 305–329, 1991.

[Mit96] John Mitchell. *Foundations for Programming Languages*. MIT Press, Cambridge, MA, 1996.

[MTW87] Bernhard Möller, Andrzej Tarlecki, and Martin Wirsing. Algebraic specifications of reachable higher-order algebras. In Donald Sannella and Andrzej Tarlecki, editors, *Recent Trends in Data Type Specification, 5th Workshop on Abstract Data Types, Gullane, Scotland, September 1-4, 1987, Selected Papers*, volume 332 of *Lecture Notes in Computer Science*, pages 154–169. Springer, 1987.

[PA93] Gordon D. Plotkin and Martín Abadi. A logic for parametric polymorphism. In *International Conference on Typed Lambda Calculi and Applications (TLCA) , Utrecht, The Netherlands*, volume 664 of *Lecture Notes in Computer Science*, pages 361–375. Springer-Verlag, 1993.

[PD06] Marius Petria and Răzvan Diaconescu. Abstract Beth definability in institutions. *Journal of Symbolic Logic*, 71(3):1002–1028, 2006.

[Pit00] Andrew M. Pitts. Parametric polymorphism and operational equivalence. *Mathematical Structures in Computer Science*, 10(3):321–359, 2000.

[Plo80] Gordon Plotkin. Lambda-definability in the full type hierarchy. In *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 365–373. Academic Press, 1980.

[Rey74] John C. Reynolds. Towards a theory of type structure. In *Paris colloquium on programming*, volume 19 of *Lecture Notes in Computer Science*. Springer, 1974.

[Rey83] John C. Reynolds. Types, abstraction, and parametric polymorphism. In *Information Processing '83*, pages 513–523. North-Holland, Amsterdam, 1983.

[RG00] Grigore Rosu and Joseph Goguen. Circular coinduction. In *In International Joint Conference on Automated Reasoning*, 2000.

[Ros03] Grigore Rosu. Inductive behavioral proofs by unhiding. *Electronic Notes in Theoretical Computer Science*, 82(1), 2003.

[RP90] John C. Reynolds and Gordon D. Plotkin. On functors expressible in the polymorphic typed lambda calculus. In Gérard Huet, editor, *Logical Foundations of Functional Programming*, University of Texas at Austin Year of Programming, pages 127–152. Addison-Wesley, Reading, Massachusetts, 1990. A preliminary version of a paper to appear in Information and Computation.

[San99] Donald Sannella. Algebraic specification and program development by stepwise refinement. In *LOPSTR*, volume 1817 of *Lecture Notes in Computer Science*, pages 1–9. Springer, 1999.

[SB83] Donald Sannella and Rod Burstall. Structured theories in LCF. In Giorgio Ausiello and Marco Protasi, editors, *Proceedings of the 8th Colloquium on Trees in Algebra and Programming*, volume 159 of *Lecture Notes in Computer Science*, pages 377–391. Springer, 1983.

[Sch87] Oliver Schoett. *Data Abstraction and the Correctness of Modular Programs*. PhD thesis, University of Edinburgh, Department of Computer Science, 1987.

[ST87] Donald Sannella and Andrzej Tarlecki. On observational equivalence and algebraic specification. *Journal of Computer and System Sciences*, 34:150–178, 1987.

[ST88a] Donald Sannella and Andrzej Tarlecki. Specifications in an arbitrary institution. *Information and Computation*, 76(2/3):165–210, 1988.

[ST88b] Donald Sannella and Andrzej Tarlecki. Toward formal development of programs from algebraic specifications: Implementations revisited. *Acta Informatica*, 25:233–281, 1988.

[STar] Donald Sannella and Andrzej Tarlecki. *Foundations of Algebraic Specification and Formal Program Development*. Cambridge University Press, to appear.

[Str67] Christopher Strachey. Fundamental concepts in programming languages. Lecture Notes, International Summer School in Computer Programming, Copenhagen, aug 1967. Reprinted in *Higher-Order and Symbolic Computation*, 13(1/2), pp. 1–49, 2000.

[Tar86] Andrzej Tarlecki. Bits and pieces of the theory of institutions. In David H. Pitt, Samson Abramsky, Axel Poigné, and David E. Rydeheard, editors, *Proceedings of the Tutorial and Workshop on Category Theory and Computer Programming*, volume 240 of *Lecture Notes in Computer Science*, pages 334–360. Springer, 1986.

[Wad89] Philip Wadler. Theorems for free. In *Functional Programming Languages and Computer Architecture*, pages 347–359. ACM, 1989.