signatures

UNIVERSITÀ DEGLI STUDI DI PISA
DIPARTIMENTO DI INFORMATICA

DOTTORATO DI RICERCA IN INFORMATICA
Università di Pisa-Genova-Udine

Ph.D. Thesis: TD-4/93

Relationships between Logical Formalisms

Maura Cerioli

Abstract. This thesis provides rigorous tools to state the independence of results and specifications from the algebraic frameworks chosen to describe them.

To translate specifications and results, a new kind of formalism (institution) homomorphisms, called *simulations*, is defined.

The main applications of simulations are three: the investigation of the relationships between the expressive power of different institutions, the translation of logical tools and the definition of specification building languages that allow basic theories to be defined in several formalisms, generalizing the concept of implementation by relating specifications from two frameworks. Recent results in the theory of both partial and non-strict algebras are disseminated through the thesis as well.

March 1993

ADDR:Corso Italia 40,56125 Pisa,Italy. TEL:+39-50-510111 - TLX:590291 DIPISAI -
E_MAIL: cerioli@cisi.unige.it

Preface

Formal specifications are widely considered an important tool in software specification, design and implementation. As a result both of theoretical investigations and of preliminary attempts at applications, a variety of specification formalisms have been considered. This fragmentation of frames is conflicting with some essential requirements of any specification formalism, i.e. the ability of supporting modularity and refinement, which are clearly related to the problem of reuse of specifications. The central theme of this thesis is the independence of specifications and results from the logical formalisms in which they are formulated.

A concrete motivating example of the need for a formal tool allowing translation and comparison of specifications defined in different frameworks is the debate about the algebraic specification of partial (higher-order) functions. Indeed the relevance of such a problem is made obvious by the wide use of partial operations in programming languages and their data types and not only the scientific community didn't agree on what formalism is better, but also the criteria for such a judgment are still under investigation.

After presenting a parade of different formalisms proposed to define partial functions and in particular some recent results about the partial paradigm, the concept of *simulation* of a framework by another is introduced, adopting the notion of institution as a synonym for logical formalism. The basic idea of simulation is encoding the syntax, i.e. signatures and sentences, of a new frame by that of an already known formalism in a way consistent with the semantics, in order to transfer back results and tools.

Then simulations are used to investigate the relationships between frames, first using the relationships between specifications of partiality in different formalisms as a guide, and then applying the same technique to the analogous problem of defining non-strict functions (like the ubiquitous *if_then_else*). Three levels can be distinguished (and formalize using simulations): the"set-theoretic", where the individual models are related disregarding their categorical and logical interconnection, the "categorical", where the relation is between the categories of models, and the "logical", where the relation is between specifications (or theories).

Moreover a general categorical construction is introduced, that allows "borrowing" the logical tools of a framework to enrich another framework lacking them, provided that an appropriate mapping between the frameworks exists. This technique applies, in particular, to the translation of inference systems along simulations (in a way that soundness and completeness are preserved) and of proofs along maps of the underlying institutions (entailment systems). An instance of this technique builds an equationally complete inference system for the partial higher-order specifications starting from an equationally complete inference system for the first-order case.

Finally the notion of *institution independent metalanguage*, by Sannella and Tarlecki, is refined in *simulation independent metalanguage*, allowing specification defined in different institutions to be assembled to build new specifications, and the concept of implementation is generalized, involving models in two institutions; in this way specifications may be related that belongs to different frames.

Contents

Introduction

Most software systems needed to solve concrete problems are far too large to be handled by human minds without the support of a rigorous methodology. This was dramatically brought to the word attention, at the end of the sixties, by the so called *software crisis*. Since then the need for formal software specifications has been widely recognized.

Formal specifications, providing tools for modularity and refinement (see e.g. [49, 72]), facilitate reuse and maintenance of produced software and allow, by rapid prototyping and formal verification methodologies, keeping under control the correspondence between the still developing programs and the customer requests.

In this frame the algebraic approach finds place as the natural abstraction of the data type concept, seen as set(s) of values and operations on them, that are the only acceptable tools to manipulate the data, whose actual realization should be hidden from the user of the data type. Thus a (concrete) data type (or a program) is a (many-sorted) *algebra* on a *signature* representing the data type (or module) interface by the names classifying the values and the operation symbols on them (see e.g. [42, 41]). Symbols in the signature are used to axiomatize the basic properties and behavior of the data type, so that a user does not need to know the actual realization of the data type, but can work on the axiomatic description, that is common to every implementation of the data type. Moreover a *standard realization* exists for each data type that is characterized by two properties: *no-junk*, i.e. every value of the data type is denoted by (at least) a term built on the signature symbols, and *no-confusion*, i.e. two terms denoting the same value in this algebra denote one value in any algebra satisfying the axioms, too. These properties are strictly related with the induction principles, and in particular with proof-theoretic properties (see e.g. [64]), and correspond to *initiality*.

In the pioneering papers of the ADJ group equational logic was chosen to axiomatize the properties of the data types, following a well-established tradition in universal algebra. Indeed the study of classes of algebras definable by equations dates back to the thirties, with the works of Birkhoff resulting in his famous theorem, that states that equations characterize exactly varieties, i.e. classes of algebras closed w.r.t. subalgebras, products and quotient (see e.g. [19]). Although varieties, or their weakened notion of *quasi-varieties*, are still in vogue because of their elegant and well-established mathematical counterpart, especially if the initial approach is considered, as a result both of theoretical investigations and of preliminary attempts at applications, the nature of the algebras, the logics used to define the classes of admissible realizations of a data type and even the notion of signature have been worked out again more or less recently, producing a considerable proliferation of specification formalisms.

From a pragmatic point of view, the existence of a number of different formalisms is pretty reasonable, because each one of them may be the more comfortable to work in, depending on the problem under examination, the field tradition, the available tools and (not least) the personal taste. However in practice (and indeed this thesis started out of a bunch of concrete problems arising in the field of the algebraic specification of concurrent systems, that are sketched later) this fragmentation leads to three orders of difficulty:

- a number of analogous results are proved in different frameworks, with slight variations of one proof technique; think for instance of the proofs of existence of an initial object in classes of (partial, total, order-sorted one/many-sorted) algebras (with/without predicates) defined by a set of conditional sentences. The standard technique is to define congruences in a way that the kernels of the natural interpretation of terms is a congruence and then show that the quotient of the term algebras by the intersection of the kernels of the natural interpretation of terms in the algebras of the class (alternatively by the congruence generated by a Birkhoff-like deduction system) is a model, too. Thus a number of instances of the same result, proved with the same technique are disseminated in the literature.

- it is virtually impossible to formally compare results and tools developed in different frameworks without the help of a meta-formalism; thus quite often comparisons and discussions about what framework better fits the needs are developed at an informal level and the lack of rigour comes out in a lack of objective criteria to establish the merits of the different frames.

- from the point of view of a specification language user the ability of supporting modularity and refinement are essential in order to allow reusing of specifications. Thus it is important (not to say crucial) to assemble, possibly at different levels of implementative detail, specification modules in different formalisms. Rephrasing the title of a landmark paper by Burstall and

> Goguen [27], the issue is "putting together theories *from different formalism* to make specifications".

To overcome these problems the first step is developing a theory whose objects are the specification frameworks, so that results can be stated in a general form and then instantiated on an actual framework; moreover such a notion should help in comparing different formalisms providing a common language to define which results/features are relevant w.r.t. the comparison. In Mahr & Makowsky's words ([56]), the goal is

> to create an appropriate framework to speak about specification languages, to compare their expressive power and to axiomatize their semantic behavior. This framework is meant to capture only the basic properties which are satisfied by any specification language that is reasonable.

Starting from ground work in the field of specification languages and in particular to give the semantics of the Clear language (see e.g. [28]), in the late seventies Burstall & Goguen captured the intuition of the minimal requirements that a theory must meet to be a reasonable specification framework and introduced the notion of *institution* "to do the Clear tricks once and for all, over any (suitable) logical system" (Burstall & Goguen [45]). Institutions formalize the intuitive notion of *logical system* and consist of the minimal ingredients of any algebraic framework:

- a collection of the admissible languages (that in the algebraic tradition are called signatures);
- for every language the class of the admissible structures on that language (algebras on the signature)
- for every language a set of sentences on the language, used to axiomatize classes of structures, or models.

A *satisfaction* relation between structures and sentences on the same language is needed in order to define (basic) specifications, i.e. to axiomatize classes of models; moreover, in order to build more complex specifications starting from simple ones, the admissible changes of language are used. Accordingly sentences are translated by language changes (by the rename of the language symbols, in the standard cases) and any model on the target language can by *reduced* to a model on the source of a language change (by interpreting each source symbol as the interpretation in the model of its image along the language change). A crucial point is that, following the intuition that the language changes do not have any semantic meaning, *truth is invariant under change of notation.* From a technical point of view this property is also needed to have that the construction of specifications using such changes of notation is sound.

Translating the above discussion in categorical terms, an *institution* consists of a category of *signatures*, a couple of functors, respectively giving the set of *sentences* and the category of *models* for any given signature, and a satisfaction relation between models and sentences on the same signature such that for each signature morphism σ a model satisfies the translation of a sentence along σ iff the reduct of the model along σ satisfies the sentence (i.e. the satisfaction relation is an *extranatural transformation*).

Few alternative notions have been proposed, at a time with institutions or later, aimed to deal with problems related to specification languages, but not designed to represent algebraic formalisms in their general form, so that their applications have been local to a problem or a working group and missed the resonance of institutions. Consider for instance the *(algebraic) specification languages* by Mahr & Makovsky, independently defined during the same period as institutions and very close to this notion; they were used in [56, 55, 57] to state an important result in the Birkhoff tradition, that, roughly speaking, can be rephrased as "the specifications uniformly admitting initial object are all and only those that can be expressed by conditional axioms". But, to the author's knowledge, specification languages were not used for any other task and the following works on the subject were stated in the institution framework (see e.g. [89, 90, 92]). Analogously the model-theoretic approach by Barwise (see e.g. [15]) had no followers among the computer scientists, because some conditions were too restrictive to capture relevant examples in this field.

On the other hand structures were proposed to deal with aspects of the algebraic specifications ignored by the institutions, namely the need for the capability to represent many-valued logics, in order to capture examples from data base and knowledge theory or simply the concept of evaluation, (see e.g. [61]) and tools to speak of deductions, proofs and such (see e.g. [36] for a formalism presented as an alternative to institutions and [63] for a completion of the institution framework with logical instruments). The original definition of institution inherited from these alternative representations of logical systems and from the feed-back coming from their use by the scientific community some slight changes and generalizations (see e.g. [45]).

Since their definition, institutions were favored by the scientific community and a number of classical notions, problems and results from algebraic specifications were rephrased and approached in this more general frame; consider, indeed, e.g. [89, 90, 92] for Birkhoff-like studies on the relationships between initiality, freeness, varieties and the form of the axioms, [32] for a generalization of the module algebra concept to the institution frame, [84, 82] for specification building languages that are uniformly interpreted on any institution, [83] for lifting the notion of observational equivalences to work on a generic logical framework, [91] for rephrasing classical logic theorem and, on a more applicative side, [16], where an institution is presented to represent implementations, and [31, 78, 79], where some institutions for the specification of the abstract requirements of concurrent systems are proposed and analyzed.

The common denominator of these papers is the emphasis on the *institution independence* of many classical results, that can be presented in a more general form once and for all.

Although a great part of algebraic theory can be rephrased in terms of a generic institution (possibly satisfying some extra conditions), there are constructions (or theoretical proofs or automatical tool building) that cannot be done without fixing the logical framework details, for example the nature of signature should be known to the designer of a parser. Moreover it may be convenient (and in practice this happens in many cases), in order to develop different parts, to be able to work in more than one institution at one time. To make this idea precise and to inherit from the category theory predefined concepts (like the notion of sub-institution) and constructions (like products, sums, or (co)limits) a notion of *arrow* between institutions has to be introduced.

The first attempt at the definition of such arrows was the notion of *institution morphism* in the pioneering paper on institutions [44]. This definition serves the purpose of building new institutions starting from simpler ones, adding in a modular way features to logical systems. The basic idea is that signature and models are translated from the source into the target institution, while sentences are mapped from the target into the source, preserving the satisfaction relation in the sense that the translation of a model satisfies all and only those sentences whose translations are satisfied by the model itself.

Although institution morphisms induce a category of institutions, where standard categorical construction correspond to intuitive institution building operations (see e.g. [44, 46] and the study of categorical properties in [91]) and are apparently well behaving also w.r.t. more *ad hoc* hierarchic constructions (the analysis of such compatibility is still under development, see [30]), they are in-appropriate to compare the expressive power of different institutions[1]. Moreover, since signatures and sentences are mapped in the opposite directions, the use of institution morphisms in order to translate axiomatic presentations of specifications are innatural.

The analysis of a few concrete examples leads to identify two main requirements to capture the notion of comparison between the logical expressive power of different institutions:

- signatures and sentences should be mapped in the same direction, so that axiomatic presentations of specifications can be translated and compared with theories expressed in a different institution;
- requiring that every (target) model may be translated (into a source model) in a way that satisfaction is preserved (with the same meaning as for institution morphisms), i.e. that every individual model of the target institution is a sound representation of a model of the source, is too restrictive, because in this way only institutions whose models have the same expressive power could be related.

Moreover, in order to be guaranteed that every specification in the source has a corresponding specification in the target whose models represent the models of the specification (where specification stands for class of algebras on one signature, as necessary to deal with specification languages), the surjectivity of the model component is crucial.

In order to meet the second requirement, there are basically three possibilities: the component dealing with models is partial (adopted by *simulations* in [2]), a signature is translated into a theory whose models are the domain of the component dealing with models (adopted by *maps of institutions* in [63]), and a (source) model is represented by a class of (target) models (adopted by *institution coding* in [93], and more recently by *pre-institution transformation* in [81], where every source model is translated into a set of target models).

Starting from a number of concrete examples and following the requirements described above, in [2] *simulations of institutions* were defined and used to approach two main problems: how to put together specifications defined in different institutions and how to generalize the notion of implementation, relating models in different formalisms.

Simulations consist of a mapping of signatures and sentences from the source into the target institution and of a backward translation of models preserving

[1]The motivations of this inadequacy are clearly presented in [91] and appear patently from concrete examples

satisfaction, that is *partial* and *surjective*. The domain of the model class is not required to be the model class of a set of old sentences, not even a full subcategory of the old models, in order to be able to use simulation to represent relationships between institutions that are known in literature, as for instance the representation of partial by total algebras adding (for each sort) a constant denoting the "undefined elements".

In the literature it is often claimed that a frame is *equivalent* to another one, usually in the sense that both solve the same kind of problems, or that in both the results are equivalently (un)satisfactory. But the meaning of equivalence is usually not formally defined and quite often used to denote different levels of relationship.

Indeed three levels can be distinguished and formalized by means of simulations, depending on whether the correspondence is between models, or categories of models, or specifications (theories). At the *set-theoretic* level, every model of the new frame is represented by a model in the old frame, that satisfies the same formulas, or, more precisely, corresponding formulas. This corresponds to requiring that a simulation exists from the new into the old frame (s.t. the domains of the model component is a, possibly non-full, subcategory of the old models). At this level most properties are missing, in particular no structured way of defining models is guaranteed to be preserved, because it usually involves categorical constructions. To have a *categorical* correspondence between two frames, at least the domain of the simulation has to be a full subcategory of the old models; moreover some more properties have to be required depending on the categorical structures that are intended to be preserved. Here the focus is on the initial structures and minimal conditions are given to preserve initiality. Even if there is a categorical simulation, the power of the specification languages in the two frames can be quite different; in particular it is possible that in the new frame some categories are definable by sets of sentences that are not so in the old one (and vice versa). To guarantee that the relationship is at the *logical level*, i.e. for every specification (i.e. the class of models which satisfy a set of sentences) in the new frame there exists a specification in the old frame equivalent to the given one in the categorical sense, it must be required not only that the domain of the model component is a full subcategory of the category of old models, but also that it is described by a set of old sentences.

Simulations can be used not only to compare the expressive power of different formalisms, but also to translate results and tools from the target into the source institution. Consider indeed, as an example, the mapping of inference systems. Given an inference system in the target institution an inference system for the source institution can be built, which consists of a preprocessing (the coding of both the premises and the consequence in terms of target sentences), the application of the given system and possibly a post-processing (the decoding of the answer). Since the validity of sentences is preserved by simulation and every new model is represented by at least an old one, soundness is preserved by this construction. Moreover if the domain of the simulation component dealing with models is the model class of a set of sentences, i.e. the simulation is a *map of institutions* too, also completeness is preserved. In particular this technique is applied to the simulation of partial higher-order specifications by partial first-order ones, after presenting an equationally complete inference system for the first-order case.

This construction is a particular case of a general categorical construction, that allows "borrowing" the logical tools of a framework to enrich another framework lacking them. In order to apply this construction, one needs appropriate mappings between the frameworks. After explaining the key categorical properties on which the construction rests and giving an overview of concepts in general logics, this technique is applied to few concrete examples to illustrate how it can be generally used to translate proofs along maps of the underlying institutions (entailment systems) and how to built a logical system for any given institution.

Finally the basic notion of simulation between institutions is extended to deal with the problem of structuring specifications in different formalisms, introducing the concept of *simulation independent metalanguage*, a generalization of notion of *institution independent metalanguage*, by Sannella and Tarlecki. Moreover, in connection with the refinement problem, it is shown how simulation adds a third dimension to the well known horizontal and vertical composition of implementations, thus allowing the composition of software modules not only from different formalisms, but also at different levels of abstraction.

Thesis Summary

The roots of this thesis date back to concrete experiences (of the supervisor and his group but not involving the author) in the field of algebraic specification of concurrent systems from two points of view.

On one side in order to specify concurrent calculi, higher-order partial functions are needed (see e.g. [13]), as well as to specify programming languages and their data types (see e.g. [11]; thus the need for a rigorous foundation of the theory of partial functions led to the study of partial higher-order specifications. Since partial higher-order specifications reduce (at least for term-generated models) to a first-order (infinitary) logic more powerful than usual, foundations for such a logic had to be provided first. The results about the categorical properties of model classes, published in [1, 7, 29], are summarized in the second chapter, while the equationally completeness of an inference system, presented in [1], restricted to ground deduction, and in [7, 29], in a more general form, is proved in the forth chapter. Moreover the application of these studies to the higher-order case, presented in [3, 6], are summarized, too, in the second (categorical results) and forth (logical system) chapters.

On the other side, as concurrency features are in a sense orthogonal to the algebraic formalism adopted in order to represent the data types of the language (including processes), provided that tools to represent the transition predicates are available, the SMoLCS paradigm was rephrased on different algebraic frameworks, tuning it to the particular applications; for example in [10, 13, 12] the algebraic formalism underlying the SMoLCS construction is the (plain) partial algebras, while in [14] total algebras with predicates are used and in [9] partial algebras with predicates. Since every change of the underlying formalism requires, in order to be rigorous, formal restatement of definitions and results that do not involve the concurrent features and are, hence, semantically immaterial, the need for a translator of algebraic framework that allowed to change the underlying formalism without affecting the concurrent constructions resulted in the study of simulations, that are the central theme of this thesis. Simulations were introduced in [2] and used to deal with two important aspects of specification languages: modularity and implementation; such results are reported in the last chapter. In [8] a rigorous analysis of the different levels of relationships between logical formalisms is developed by means of simulation; in the second chapter these levels are illustrated and the technique is applied to two paradighmatic cases: the relationship between different specifications of partiality and the comparison between the specification of non-strict functions in total and non-strict frames.

The structure of the thesis is the following.

The first chapter is devoted to the introduction of the notion of institution as rigorous counterpart of specification framework and to the comparison of institutions with other formalisms introduced in litterature to represent logical frames: Pre-Institutions, by Scollo and Salibra; Specification Logics, by Erigh, Baldamus and Cornelius; Galleries, by Mayoh; Foundations, by Poigné; π-institutions, by Fiadeiro and Sernadas; Entailment Systems, by Meseguer.

In the second chapter, as a paradigmatic example of the need for a rigoruos definition of specification formalism translator, the specification of partial functions in several styles is proposed, considering from the classical total ones, like *error algebras*, to more recent proposals, like the order-sorted paradigm or the use of explicit typing mechanisms. A large room is given to the partial approach, not only recalling the classical result but also proposing a treatment of higher-order partiality.

The third chapter presents the central definition of this thesis, i.e. the notion of simulation and uses simulations to enlighten the differences between a number of representations of partiality by total algebras, showing that three levels can be distinguished, corresponding to relating individual models, categories of models, or axiomatic presentations. The same technique is applied also to the relationship between total and non-strict algebras, after a summary of the results about non-strict don't care specifications. Then simulations are compared to other notions of arrows between institutions: institution morphisms, maps of institution, pre-institution transformations and istitution coding/representations.

A general categorical construction allowing a structure to "borrow" logical tools from a richer one along a mapping is presented in the forth chapter and applied to the world of general logic. In particular it is shown how to endow an institution with an entailment system via a simulation or a map of institution and this technique is applied to build an inference system for the partial higher-order framework, after presenting an equationally complete inference system for partial (first-order) conditional specifications.

The last chapter is devoted to the use of simulations to translate (structured) specifications in order to assemble modules defined in different frames by means of *simulation independent* metalanguages, that are a refinement of the institution independent metalanguages introduced by Sannella and Tarlecki. Moreover simulations are used to generalize the notion of implementation (as refinement), by allowing specifications defined in different institutions to be related and adding, in this way, a third dimension to the horizontal and vertical compositions.

Acknowledgments

As the modularity principle is the main stream of this thesis, it would be unreasonable for the acknowledgments to be unstructured; on the other hand many people should appear in different subsections, like Egidio Astesiano, that deserves my thanks as teacher, for introducing me to the theoretical computer science, as supervisor, for its efforts in order to improve my expressive capabilities and my scientific autonomy, and especially as colleague and friend, for its continual support and availability. Thus I'll try to pass between *Scilla* and *Cariddi*.

Friends, Relatives and such

Since I always regarded love and friendship as the more important and basilar needs for human life (or at least for *my* life), let me start this long list of people that positively influenced and helped me, with the emotional supporter.

Relatives

First of all, at least for chronological reasons, I would like to thank my parents, my dad (have some more coin for the coffee machine) Paolo and my mom (take one more bite) Mimma, official sponsors of my studies and apparently still thinking at me as I would be teenage, that I found greatly relaxing in a world asking me for every time more seriousness and maturity. They helped and are still helping me with all their strength, showing their love in a thousand little (and big) things and making me feel that I'll have forever a safe port from any storm in my life.

I would also like to thank my sister Paola and my brother (in law[2]) Angelo, who follow with interest my life and are always at hand to discuss, suggest or simply have fun.

My special thanks are due to my husband, Massimo, for a thousand good reasons, not least for (still) being my husband in spite of these last months of my thesis development (and first of our marriage).

Friends

I'm in great doubt if I should thank Alessandro Garibbo here or in the above section. Actually we are friends from so much time that he is like a brother to me and as every brother he is never hesitant about saying me when I'm wrong, and for this endowment especially I appreciate him.

People from my working group deserves all my gratitude for the friendly way and naturalness they accepted me with, and in particular my special thanks go to[3] Elena Zucca, my office-mate, who's always available to discuss both technicalities and human matters, Gianna Reggio, Alessandro Giovini, Gerardo Costa, Egidio Astesiano, our (more friend than) boss, and to Ombretta Arvigo, our part-time secretary, whose complete availability makes possible the impossible deadlines and whose warm humanity and ever-present humor greatly improve the office atmosphere.

Last but not least, Claudia Fassino and her husband Stefano Pasquero deserve my thanks for a long and fruitful friendship.

[2] I met Angelo for the first time less than 10 years ago, but is now a brother to any extent

[3] order is immaterial

Scientific Community

Coming to the technical side, I would like to stress that, due to the friendly and open-minded attitude of the scientists I met, people listed here could be as well be classified under "friends". Thus consider the following thanks to be for the scientific help as well as for the human support.

Teachers

Just few words to thank my high-school mathematics professor, Gabriella Ferrari Nardi Greco, who thought and taught that mathematics was more than calculi and made me love the argument, and my high-school literature professor, Aldo Bartarelli, who taught me the most important thing, i.e. how to study and how to enjoy the learning process.

Computer Scientists

First of all I would like to thank the numerous scientists disseminated in Europe and U.S.A. who devoted some of their (precious) time to exchange ideas with me for the fruitful discussions about the arguments in my thesis, and in particular M. Wirsing and (his student and my friend) B. Reus, J. Goguen and his group in Oxford, that I had the pleasure to visit, D. Sannella, A. Tarlecki, H. Reichel, J. Meseguer and his group at the S.R.I. (not only for the scientific discussions, but also for the friendly welcoming in California[4]) and in particular F. Martí-Oliet, and E. Moggi.

My special thanks to the DISI people and in particular to my working group for the stimulating discussions and to Marco Grandis and Marco Borga of the Mathematics Dept. for their help respectively on category theory and logic.

Last but first my gratitude is due to Egidio Astesiano, my supervisor, who introduced me to the research, leading my interests to the computer science and supporting me for years with friendly encouragements.

LaTeX Wizards

Besides Ombretta Arvigo and Gianna Reggio, who technically speaking are not wizards, but helped me every time I needed (or at least tried to), I must thank the unique true *guru* Alessandro Giovini, who not only knows the Paths of the Faith

[4]thanks also to Daniela Musto, Lorenza Moro, Elisabetta Canavesi, Susan Kelleher and Don *I-never-get-his-family-name* who helped me on the practical side and made my visit a vacation (at least at week-end time)

and the Word of Truth about TEX and LATEX (and disseminates his enlightenment on us humble people), but with his holy presence make the computers behave (it is well known that strange phenomena stop to occur provided that the system manager is present).

I'm also in debt with Paul Taylor for his "Commutative Diagrams in TEX", that produced the diagrams in my thesis with a minumum of pain (before founding up the existence of such a package I firmly decided to hand-draw any figure) and L.Botway and C. Biemesderfer who patiently compiled a LATEX Command Summary, that proved to be essential for quick finding of the key-words.

Chapter 1

Specification Formalisms

It is widely recognized the importance of the algebraic approach as a uniform way to describe both data types and programs with natural techniques to structure and refine the design of modules.

The basic intuition of any algebraic approach consists in describing data structures (or programs) by simply fixing the *signature*, i.e. the names for the different sets of involved data and for the basic operations that build/manipulate the data, and axiomatizing their characteristic properties by means of a set of sentences (mainly Horn-clauses) built on the signature.

The semantics of an algebraic specification is then given as a class of *algebras* on the signature of the specification, i.e. of structures where each name of data type has been interpreted by a set and each name of operation by a function. In order to define such a class some machinery is common to the different algebraic approaches:

- the notion of "satisfaction" or "validity" has to be defined, in order to restrict the algebras to the ones that satisfy the axioms;

- terms on the signature are inductively defined, starting from the symbols for constants and variables and closing under the application of operation symbols and then the *natural interpretation*, or *evaluation* of terms in an algebra under a valuation for their variables is inductively defined, by interpreting the variables by their valuation and each operation symbol (including constants) by its interpretation in the algebra;

- functions are defined between algebras, that preserve the evaluation of terms and are usually called *homomorphisms*.

Then, using terms and homomorphisms, there are three basic approaches to give the semantics of a specification; the first two correspond to and generalize the mathematical principles of induction and observational/behavioural abstraction:

initial approach: the class of algebras described by the specification is the isomorphism class of the structure characterized by the two condition of *no-junk* (each element is denoted by a term on the signature) and *no-confusion* (two terms are equal iff they can be proved equal from the axioms); this corresponds to the existence of exactly one homomorphism from this structure into each algebra satisfying the axioms;

terminal approach: the class of algebras described by the specification is the isomorphism class of the structure characterized by two condition: no-junk and "two terms are equal iff they cannot be proved different from the axioms"; this corresponds to existence of exactly one homomorphism into this structure from each algebra satisfying the axioms and some extra conditions, like no-junk, strictly depending on the actual algebraic formalism;

loose approach: the specification describes the class of the algebras satisfying its axioms.

Although in the final stage of a specification process either the initial (and I mostly prefer this because of computability considerations) or the terminal approach is usually adopted in order to define exactly one structure (up to isomorphism), in the intermediate phases, when the specification is under refinement, the loose approach is the most favorite, because it allows to progressively restrict the class of models by fixing decisional details.

It is worth noting that, although the notion of term and natural interpretation is crucial in order to define axioms with their validity and homomorphisms, *a posteriori* axioms and homomorphisms are sufficient to qualify the initial, the terminal and the loose approach. Thus terms can be regarded as an auxiliary concept used to express the main notions of axiom and homomorphism, that can be dropped in a general theory to deal with specification formalisms, provided that the models have a categorical structure, by adopting the concepts of axiom and validity as primitive.

The last ingredient of any algebraic formalism is the capability of structuring specifications to modularly build large specifications. In order to put together specifications, a *specification language* is needed and although many different languages have been defined in the last years, all share the use of renaming of the signature symbols in order to relate basic specifications and in particular to instantiate parametric specifications. In this phase it appears crucial that the change of notation does not affect the validity of sentences, in order to have that the models of a larger specification, restricted to the symbols of any its subcomponent are models of this subcomponent, too.

In order to have a formalism able to relate and/or put together specifications expressed in different frameworks, a precise notion of what a specification framework is has to be provided. This chapter is devoted to the introduction of the notion of *institution* (see e.g. [44, 45, 46]), starting with the paradigmatic example of the many-sorted framework, and the comparison of this concept to alternative formalisms.

1.1 Institutions

As introductory example of specification formalism, the probably most famous is considered: the many-sorted algebras.

Signatures consist of a set of names for the data types and a family of function symbols together with their functionality and the signature morphisms are the renaming of names that preserve the functionality of operation symbols.

Def. 1.1.1 A *many-sorted signature* (S, F) consists of a countable set S of *sorts* and of a family $F = \{F_{w,s}\}_{w \in S^*, s \in S}$ of disjoint sets of *operation symbols*. In the sequel $op: s_1 \ldots s_n \to s$ stands for $op \in F_{s_1 \ldots s_n, s}$. A generic signature will be denoted by Σ.

Let $\Sigma_1 = (S_1, F_1)$ and $\Sigma_2 = (S_2, F_2)$ be many-sorted signatures; then a *signature morphism* $(\sigma, \phi): \Sigma_1 \to \Sigma_2$ consists of a *sort renaming* $\sigma: S_1 \to S_2$ and a family $\phi = \{\phi_{w,s}: F_{1w,s} \to F_{2\sigma(w),\sigma(s)}\}$, where $\sigma(w)$ is inductively defined by $\sigma(\Lambda) = \Lambda$ and $\sigma(sw) = \sigma(s)\sigma(w)$. □

The algebras on a many sorted signature are the structures that associate with each sort a carrier set and to each operation symbol a concrete function with the proper arity; the allowed morphisms between two such structures are the mapping of the supporting carriers that preserve the interpretation of function symbols.

Def. 1.1.2 A *many-sorted algebra* A on a signature $\Sigma = (S, F)$ consists of a family $\{s^A\}_{s \in S}$ of sets, the *carriers*, and of a family $\{op^A\}_{op \in F_{w,s}, w \in S^*, s \in S}$ of functions, the *interpretations of operation symbols*, s.t.

- if $w = \Lambda$ then $op^A \in s^A$;
- if $w = s_1 \ldots s_n$, where $n \geq 1$, then $op^A: {s_1}^A \times \ldots \times {s_n}^A \to s^A$.

Often the algebra A is denoted by the couple $(\{s^A\}, \{op^A\})$, omitting the quantifications about s and op which are associated with the signature. A many-sorted algebra over a signature Σ is called a Σ-algebra. The class of all Σ-algebras is denoted by $Alg(\Sigma)$.

In order to unify the notation, in the sequel the constants $op \in F_{\Lambda,s}$ are regarded as "zeroary" operations, with the convention that $s_1 \dots s_0$ stands for Λ and that $\Lambda^A = {s_1}^A \times \dots \times {s_0}^A$ is the singleton set, so that a function having it as domain is an element of the codomain, accordingly with the definition of op^A.

Let A and B be Σ-algebras. Then a *homomorphism* p from A into B is a family of functions $p = \{p_s : s^A \to s^B\}_{s \in S}$ s.t. for any $op \in F_{s_1 \dots s_n, s}$, with $n \geq 0$, and any $a_i \in {s_i}^A$ with $i = 1 \dots n$,

$$p_s(op^A(a_1, \dots, a_n)) = op^B(p_{s_1}(a_1), \dots, p_{s_n}(a_n)).$$

In the sequel $p: A \to B$ will denote a homomorphism p from A into B. □

For any signature morphism $(\sigma, \phi): (S_1, F_1) \to (S_2, F_2)$ a functor $Alg(\sigma, \phi): Alg(S_2, F_2) \to Alg(S_1, F_1)$ exists called *reduct*.

Def. 1.1.3 Let $\Sigma_1 = (S_1, F_1)$ and $\Sigma_2 = (S_2, F_2)$ be signatures and $(\sigma, \phi): \Sigma_1 \to \Sigma_2$ be a signature morphism; then the *reduct* functor $Alg(\sigma, \phi): Alg(\Sigma_2) \to Alg(\Sigma_1)$ is defined by:

- for each Σ_2-algebra A, the Σ_1-algebra $Alg(\sigma, \phi)(A)$ consists of:
  - $s^{Alg(\sigma,\phi)(A)} = (\sigma(s))^A$ for all $s \in S_1$
  - $op^{Alg(\sigma,\phi)(A)} = (\phi(op))^A$ for all $op \in F_1$
- for each homomorphism h between Σ_2-algebras, the homomorphism $Alg(\sigma, \phi)(h)$ is the family $\{Alg(\sigma, \phi)(h)_s = h_{\sigma(s)} \mid s \in S_1\}$. □

A particular example of algebra is the *term-algebra*. In the sequel variables are assumed to be "new" symbols, disjoint w.r.t. the symbols used to build signatures.

Def. 1.1.4 Let $\Sigma = (S, F)$ be a signature and $X = \{X_s\}_{s \in S}$ be an S-sorted family of variables.

- The family $\{T_\Sigma(X)_{|s}\}_{s \in S}$ of the *sets of the terms* is inductively defined by:
  - $X_s \cup F_{\Lambda,s} \subseteq T_\Sigma(X)_{|s}$, for all $s \in S$;
  - $op(t_1, \dots, t_n) \in T_\Sigma(X)_{|s}$, for all $op \in F_{s_1 \dots s_n, s}$ and all $t_i \in T_\Sigma(X)_{|s_i}$ for $i = 1 \dots n$.

- For all $op \in F_{s_1 \ldots s_n, s}$ let $op^T: T_\Sigma(X)_{|s_1} \times \ldots \times T_\Sigma(X)_{|s_n} \to T_\Sigma(X)_{|s}$ be defined by $op^T(t_1, \ldots, t_n) = op(t_1, \ldots, t_n)$ for all $t_i \in T_\Sigma(X)_{|s_i}$ for $i = 1 \ldots n$.

- The *term-algebra* over Σ and X, denoted by $T_\Sigma(X)$, or shortly T_Σ if X is the (family of) empty set, is the couple $(\{T_\Sigma(X)_{|s}\}, \{op^T\})$.

- Let A be a Σ-algebra and $V = \{V_s: X_s \to s^A \mid s \in S\}$ be an evaluation of the variables in A; then the *natural evaluation* $eval^{A,V}$ of the terms in A w.r.t. V is inductively defined by:

  - $eval^{A,V}(x) = V_s(x)$ for all $x \in X_s$ and all $s \in S$;
  - $eval^{A,V}(op(t_1, \ldots, t_n)) = op^A(eval^{A,V}(t_1), \ldots, eval^{A,V}(t_n))$ for all terms $op(t_1, \ldots, t_n)$.

  In the sequel $eval^{A,V}(t)$ will be denoted by $t^{A,V}$ or simply by t^A, if X is the (family of) empty set. □

To simplify the treatment of the logical component, only conditional formulas without variables are considered and the generalization to Horn-clauses with variables will be introduced in the sequel.

Def. 1.1.5 Let $\Sigma = (S, F)$ be a signature.

- A *ground equality* over Σ has the form $t = t'$ for $t, t' \in T_{\Sigma|s}$. The set of all ground equalities over Σ will be denoted by $GEq(\Sigma)$.

- A *ground conditional formula* over Σ has the form

$$t_1 = t'_1 \wedge \ldots \wedge t_n = t'_n \supset t = t',$$

  where $t_1 = t'_1, \ldots, t_n = t'_n, t = t' \in GEq(\Sigma)$. The set of all ground conditional formulas over Σ will be denoted by $GCond(\Sigma)$.

- If A is a many-sorted algebra and ϕ is a formula, then ϕ *holds* in A (equivalently: *is satisfied by* A), denoted by $A \models_{\mathcal{GMS}} \phi$, accordingly to the following:

  - $A \models_{\mathcal{GMS}} t = t'$ iff $t^A = t'^A$;
  - let ϕ be $t_1 = t'_1 \wedge \ldots \wedge t_n = t'_n \supset t = t'$; then $A \models_{\mathcal{GMS}} \phi$ iff $A \models_{\mathcal{GMS}} t = t'$, or $A \not\models_{\mathcal{GMS}} t_i = t'_i$ for some $i \in \{1 \ldots n\}$. □

For any signature morphism $(\sigma, \phi): (S_1, F_1) \to (S_2, F_2)$ it is easy to define the translation of formulas along the signature morphism as the renaming of function symbols.

Def. 1.1.6 Let $\Sigma_1 = (S_1, F_1)$ and $\Sigma_2 = (S_2, F_2)$ be signatures and ρ be the signature morphism $(\sigma, \phi): \Sigma_1 \to \Sigma_2$

The *renaming* $ren_\rho: GCond(\Sigma_1) \to GCond(\Sigma_2)$ is defined by

- $ren_\rho(t = t')$ is $ren_\rho(t) = ren_\rho(t')$ and
- $ren_\rho(\epsilon_1 \wedge \ldots \wedge \epsilon_n \supset \epsilon)$ is $ren_\rho(\epsilon_1) \wedge \ldots \wedge ren_\rho(\epsilon_n) \supset ren_\rho(\epsilon)$

where $ren_\rho(t)$ is inductively defined by

- $ren_\rho(op) = \phi_s(op)$ for all $op \in F_{1\Lambda,s}$;
- $ren_\rho(op(t_1, \ldots, t_n)) = \phi(op)(ren_\rho(t_1), \ldots, ren_\rho(t_n))$ for all $op \in F_{1_{s_1 \ldots s_n, s}}$. $\square$

It is easy to check that for each signature morphism the translation of a sentence along the morphism is satisfied by a model iff the reduct of this models satisfies the sentence.

Abstracting the case of many-sorted algebras on the base of the above discussion, an algebraic specification formalism can be summarized by:

- a collection of signatures, together with their morphisms, corresponding to the languages to express the data types and their translations;
- for each signature a set of sentences and for each signature morphism from Σ into Σ' a renaming of sentences on Σ into sentences on Σ';
- for each signature a collection of models, or algebras, on this signature, together with their homomorphisms, and for each signature morphism from Σ into Σ' a reduct functor from models on Σ' into models on Σ;
- for each signature a satisfaction relation, relating models and sentences on the same signature.

and since the signature morphisms are just syntactical translations, satisfaction changes consistently with any signature morphism. This corresponds to the notion of *institution*, first introduced by Burstall and Goguen, to define the semantic of the language Clear.

Def. 1.1.7 [[44] def.14] An *institution* $\mathcal{I}$ consists of

- a category **Sign** of *signatures*;
- a functor Sen: **Sign** $\to$ **Set** giving the set of *sentences* over a given signature;

- a functor $Mod: \mathbf{Sign}^{op} \to \mathbf{Cat}$ giving the category (sometimes called the variety) of *models* of a given signature (the arrows in $Mod(\Sigma)$ are called the *model morphisms*);

- a satisfaction relation[1]

$$\models \subseteq |Mod(\Sigma)| \times Sen(\Sigma)$$

  for each Σ in **Sign**, sometimes denoted $\models_\Sigma$, such that for each morphism $\phi: \Sigma_1 \to \Sigma_2$ in **Sign**, the *Satisfaction Condition*

$$M' \models Sen(\phi)(\xi) \iff Mod(\phi)(M') \models \xi$$

  holds for each M' in $|Mod(\Sigma_2)|$ and each ξ in $Sen(\Sigma_1)$. □

The many-sorted formalism is easily rephrased as an institution.

Def. 1.1.8 The institution of many sorted total algebras with ground conditional formulas is the quadruple $\mathcal{GMS} = (\mathbf{Sign}_{\mathcal{MS}}, Sen_{\mathcal{GMS}}, Mod_{\mathcal{MS}}, \models_{\mathcal{GMS}})$, where:

the category $\mathbf{Sign}_{\mathcal{MS}}$ has many-sorted signatures as objects and many-sorted signature morphisms as arrows, introduced by Def. 1.1.1

The functor $Sen_{\mathcal{GMS}}: \mathbf{Sign}_{\mathcal{MS}} \to \mathbf{Set}$ consists of:

- $Sen_{\mathcal{GMS}}(\Sigma) = GCond(\Sigma)$, introduced by Def. 1.1.5, for each signature Σ;

- $Sen_{\mathcal{GMS}}(\rho) = ren_\rho$, introduced by Def. 1.1.6, for each signature morphism ρ.

The functor $Mod_{\mathcal{MS}}: {\mathbf{Sign}_{\mathcal{MS}}}^{op} \to \mathbf{Cat}$ consists of:

- $Mod_{\mathcal{MS}}(\Sigma)$ is the category of many-sorted algebras, introduced by Def. 1.1.2, for any signature Σ;

- $Mod_{\mathcal{MS}}(\rho)$ is the reduct functor, introduced by Def. 1.1.3, for any signature morphism ρ.

For any $\Sigma \in |\mathbf{Sign}_{\mathcal{MS}}|$, any $A \in |Mod_{\mathcal{MS}}(\Sigma)|$ and any $\xi \in Sen_{\mathcal{GMS}}(\Sigma)$, $A \models_{\mathcal{GMS}} \xi$ accordingly with Def. 1.1.5. □

In order to generalize the notion of sentences to allow conditional formulas with variables, some care has to be taken to deal with the translation of variables along signature morphisms. Following the approach in [46], the notation for many-sorted conditional formulas has been slightly changed w.r.t. the classical algebraic approach (see e.g. [64, 34]).

[1] for any category $\mathbf{C}$ the class of the objects of $\mathbf{C}$ is denoted by $|\mathbf{C}|$.

Def. 1.1.9 The institution of many sorted total algebras with (open) conditional formulas is the quadruple $\mathcal{MS} = (\mathbf{Sign}_{\mathcal{MS}}, Sen_{\mathcal{MS}}, Mod_{\mathcal{MS}}, \models_{\mathcal{MS}})$, where $\mathbf{Sign}_{\mathcal{MS}}$ and $Mod_{\mathcal{MS}}$ are as in Def. 1.1.8 and the functor $Sen_{\mathcal{MS}}: \mathbf{Sign}_{\mathcal{MS}} \to \mathbf{Set}$ consists of:

- for any signature $\Sigma = (S, F)$ the set $Sen_{\mathcal{MS}}(\Sigma)$ consisting of formulas

  $$V.t_1 = t'_1 \wedge \ldots \wedge t_n = t'_n \supset t = t',$$

  where $V: X \to S$ is a partial typing function and $t_i \in T_\Sigma(\{V^{-1}(s)\}_{s \in S})$.

- for any signature morphism (σ, ϕ), the function $Sen_{\mathcal{MS}}(\sigma, \phi)(V.\xi) = (\sigma \circ V.\bar{\phi}(\xi))$, where $\bar{\phi}$ is the usual rename of the function symbols leaving the variables unaffected.

For any $\Sigma \in |\mathbf{Sign}_{\mathcal{MS}}|$, any $\xi = (V.t_1 = t'_1 \wedge \ldots \wedge t_n = t'_n \supset t = t') \in Sen_{\mathcal{GMS}}(\Sigma)$ and any $A \in |Mod_{\mathcal{MS}}(\Sigma)|$, $A \models_{\mathcal{MS}} \xi$ iff for every valuation $U = \{U_s: V^{-1}(s) \to s^A\}_{s \in S}$ of the variables of ξ in A:
$A \models_{\mathcal{MS}_U} t = t'$ or $i \in \{1, \ldots, n\}$ exists s.t. $A \not\models_{\mathcal{MS}_U} t_i = t'_i$, where $A \models_{\mathcal{MS}_U} t = t'$ iff $t^{A,U} = t'^{A,U}$.

The institution of many sorted total algebras with (open) equational formulas is the subinstitution[2] $\mathcal{EMS} = (\mathbf{Sign}_{\mathcal{MS}}, Sen_{\mathcal{EMS}}, Mod_{\mathcal{MS}}, \models_{\mathcal{MS}})$ of $\mathcal{MS}$, where for every signature Σ the sentences $Sen_{\mathcal{EMS}}(\Sigma)$ are the subset $\{V.\xi \mid \xi = (t = t')\}$ of $Sen_{\mathcal{MS}}(\Sigma)$. □

First Order Structures

A very close framework to standard (one-sorted) many-sorted algebras with conditional sentences based on equality is the theory of (homogeneous) heterogeneous first-order structures (algebras with predicates), where predicates are allowed in order to build formulas and accordingly algebras (i.e. models) are equipped with the truth set of each predicate, to define validity of sentences, that, as usual in computer science, are (positive) Horn-Clauses.

Besides the theoretical relevance of this approach, predicates are useful in order to represent many central concepts in computer science, like transistions of concurrent systems and typing relations. Thus the general theory of both homogeneous and heterogeneous first-order structures is here briefly summarized.

[2]Here and in the sequel the word "subinstitution" is used rather informally to denote an institution with less signatures and/or less sentences and/or less models, without a rigorous categorical counterpart.

Def. 1.1.10 The institution of *many-sorted first-order structures* with equality is the quadruple $\mathcal{TL} = (\mathbf{Sign}_{\mathcal{TL}}, Sen_{\mathcal{TL}}, Mod_{\mathcal{TL}}, \models_{\mathcal{TL}})$, where:

- $\mathbf{Sign}_{\mathcal{TL}}$ is the category whose objects (S, F, P) consist of a many-sorted signature (S, F) and an S^+-indexed family P of *predicates* and whose morphisms $(\sigma, \phi, \pi)\colon (S_1, F_1, P_1) \to (S_2, F_2, P_2)$ consist of a many-sorted morphisms $(\sigma, \phi)\colon (S_1, F_1) \to (S_2, F_2)$ and a type preserving S^+-indexed family of functions $\pi_{s_1 \ldots s_n}\colon {P_1}_{s_1 \ldots s_n} \to {P_2}_{\sigma(s_1 \ldots s_n)}$.

- $Sen_{\mathcal{TL}}\colon \mathbf{Sign}_{\mathcal{TL}} \to \mathbf{Set}$ is defined by:

  - for every signature $\Sigma = (S, F, P)$ in $\mathbf{Sign}_{\mathcal{TL}}$ the set $Sen_{\mathcal{TL}}(\Sigma)$ is

    $$\{V.\epsilon_1 \wedge \ldots \wedge \epsilon_n \supset \epsilon \mid \epsilon_1 \ldots \epsilon_n, \epsilon \in Atoms(\Sigma, V)\}$$

    where V is a sort assignment to the variables and $Atoms(\Sigma, V)$ consists of the equalities $t = t'$ between Σ-terms (on V-sorted variables) and the *atomic formulas* of the form $p(t_1, \ldots, t_k)$ for t_i terms of sort s_i, $i = 1 \ldots k$, and $p \in P_{s_1 \ldots s_k}$;

  - for every signature morphism $\rho\colon \Sigma \to \Sigma'$, where $\rho = (\sigma, \phi, \pi)$, the translation $Sen_{\mathcal{TL}}(\sigma)$ of a sentence $V.\xi$ is $V \circ \sigma.ren(\xi)$, where $ren(\xi)$ denotes the renaming of function symbols in ξ by ϕ and the predicate symbols by π, leaving unaffected the variables.

- $Mod_{\mathcal{TL}}\colon {\mathbf{Sign}_{\mathcal{TL}}}^{op} \to \mathbf{Cat}$ is defined by:

  - for every signature $\Sigma = (S, F, P)$ in $\mathbf{Sign}_{\mathcal{TL}}$ the category $Mod_{\mathcal{TL}}(\Sigma)$ consists of:

    * the objects are *first-order structures* $(\{s^A\}_{s \in S}, \{f^A\}_{f \in F}, \{p^A\}_{p \in P})$, where $(\{s^A\}_{s \in S}, \{f^A\}_{f \in F})$ are many-sorted algebras and p^A is a subset of ${s_1}^A \times \ldots \times {s_n}^A$ for each $p \in P_{s_1 \ldots s_n}$;
    * the morphisms are *truth preserving* many-sorted homomorphisms $h\colon A \to B$ s.t.
      if $(a_1, \ldots, a_n) \in p^A$, then $(h_{s_1}(a_1), \ldots, h_{s_n}(a_n)) \in p^B$ for all predicates p.

  - for every signature morphism $\rho\colon \Sigma \to \Sigma'$, where $\rho = (\sigma, \phi, \pi)$, the translation $Mod_{\mathcal{TL}}(\sigma)(A')$ of a first order structure A' is the reduct $(\{\sigma(s)^A\}_{s \in S}, \{\phi(f)^A\}_{f \in F}, \{\pi(p)^A\}_{p \in P})$ and the translation of a morphism h is $Mod_{\mathcal{MS}}(\sigma)(h)$.

- For any $\Sigma \in |\mathbf{Sign}_{\mathcal{TL}}|$, any $\xi = (V.\epsilon_1 \wedge \ldots \wedge \epsilon_n \supset \epsilon) \in Sen_{\mathcal{TL}}(\Sigma)$ and any $A \in |Mod_{\mathcal{TL}}(\Sigma)|$, $A \models_{\mathcal{TL}} \xi$ iff $A \models_{\mathcal{TL}_U} \epsilon$ or there exists $i \in \{1, \ldots, n\}$ s.t. $A \not\models_{\mathcal{TL}_U} \epsilon_i$ for all valuations $\{U_s : V^{-1}(s) \to s^A\}_{s \in S}$, where $A \models_{\mathcal{TL}_U} t = t'$ iff $t^{A,U} = t'^{A,U}$ and $A \models_{\mathcal{TL}_U} p_n(t_1, \ldots, t_{k_n})$ iff $({t_1}^{A,U}, \ldots, {t_n}^{A,U}) \in p^A$.

The institution of *(one-sorted) first-order structures* with equality is the subinstitution $\mathcal{L} = (\mathbf{Sign}_{\mathcal{L}}, Sen_{\mathcal{L}}, Mod_{\mathcal{L}}, \models_{\mathcal{L}})$ of $\mathcal{TL}$, whose signature have singleton sets of sorts and whose sentences and models are the composition of $Sen_{\mathcal{TL}}$ (resp. $Mod_{\mathcal{TL}}$) with the embedding of $\mathbf{Sign}_{\mathcal{L}}$ into $\mathbf{Sign}_{\mathcal{TL}}$. $\square$

Since in the homogeneous case the name of the sort is immaterial, the notation is simplified dropping the sort indexes and the typing of variables in sentences.

Alternative Formulation

There is also a more categorical formulation of the concept of institution (see e.g. [45]), that is worth to be recalled, because it is not only more elegant, but also easier to generalize.

Def. 1.1.11 Let $\mathbf{C}$ and $\mathbf{B}$ be categories, $S: \mathbf{C}^{op} \times \mathbf{C} \to \mathbf{B}$ be a functor and let b be an object of $\mathbf{B}$. Then an *extranatural transformation*[3] $\alpha: S \to b$ is a function assigning to each object c of $\mathbf{C}$ a morphism $\alpha_c: S(c, c) \to b$ in $\mathbf{B}$ such that for any $f: c \to c'$ in $\mathbf{C}$ the following diagram commutes

$$\begin{tikzcd}
S(c', c) \arrow[r, "S(Id_{c'}, f)"] \arrow[d, "S(f, Id_c)"'] & S(c', c') \arrow[d, "\alpha_{c'}"] \\
S(c, c) \arrow[r, "\alpha_c"'] & b
\end{tikzcd}$$

An institution is a pair of functors $Mod: \mathbf{Sign}^{op} \to \mathbf{Cat}$ and $Sen: \mathbf{Sign} \to \mathbf{Set}$ with an extranatural transformation $\models: |Mod(\_)| \times Sen(\_) \to \{true, false\}$, where the functor $|\_|$ is the forgetful functor from $\mathbf{Cat}$ to $\mathbf{Set}$, discarding the arrows. $\square$

1.2 Other Formalisms

In the last years a small collection of alternative notions of specification formalisms has been developed, aimed to generalize the treatment of the logical side. These

[3]see e.g. [54]

different theories can be essentially grouped in three classes, each of them facing one of three possible lacks of generality from the institution theory:

- the satisfaction condition is claimed to be too restrictive, because it does not capture non-monotonic reasoning, where for example adding symbols to the language can affect the validity of sentences; although the motivation seems reasonable in principle, the only significant example that has been fully developed until now, the behavioural equational specifications, is quite controversial; indeed it is sufficient a slight change of the definition of sentences to get an institution, but it is still unexplored the correspondence between this institution and the practical use of behavioural specifications;

- by definition the satisfaction relation determines whether a sentence is, or is not, satisfied by a model; thus the concept of institution can capture just two-valued logics, while many applications need more liberal notions of validity;

- the concept of institution misses built-in tools for the treatment of deduction and proofs, so that the theory of (automatic) theorem provers can be hardly formalize using institutions.

In the sequel the main alternative formalisms are presented, starting with the *pre-institutions* by Salibra and Scollo (see [81]) and the *specification logics* by Ehrig, Baldamus, Cornelius and Orejas (see [33]), where the satisfaction condition is relaxed.

Then the *galleries* by Mayoh (see [61]) are briefly sketched, where the notion of satisfaction is replaced by the idea of "evaluation" of (possibly logical) expressions, and compared to the *generalized institutions* by Burstall and Goguen (see [45]).

Finally built-in tools to deal with logical concepts are introduced with the *π-institutions* by Fiadeiro and Sernadas (see [36]), where essentially models and validity are dropped and substituted by a *consequence relation*, and with the *entailment systems* by Meseguer (see [63]), where a notion closed to the one of π-institution is inserted in a big "fresco", including the theory of institutions, where notions to represent not only entailment systems, but also proofs and calculi are at hand and the different parts of the picture are strictly related by a net of (adjoint) functors.

1.2.1 Pre-Institutions

In order to investigate which properties of a logical formalism depend on the satisfaction condition and/or the categorical structure of the model classes, and which do not, holding for larger classes of formalisms than the institutions, recently Salibra and Scollo in [81] introduced a weakening of the notion of institution, called *pre-institution*, where both the satisfaction condition and the categorical structure of the models have been dropped.

Def. 1.2.1 A *pre-institution* is a 4-tuple $\mathcal{I} = (\mathbf{Sign}, Sen, Mod, \models)$, with:

- $\mathbf{Sign}$ is a category, whose objects are called *signatures*,
- $Sen: \mathbf{Sign} \to \mathbf{Set}$ a functor, sending each signature Σ to the set $Sen(\Sigma)$ of Σ-sentences, and each signature morphism $\sigma: \Sigma \to \Sigma'$ to the mapping $Sen(\sigma): Sen(\Sigma) \to Sen(\Sigma')$ that translates Σ-sentences to Σ'-sentences,
- $Mod: \mathbf{Sign} \to \mathbf{Set}$ a functor, sending each signature Σ to the set $Mod(\Sigma)$ of Σ-models, and each signature morphism $\sigma: \Sigma \to \Sigma'$ to the σ-reduction function $Mod(\sigma): Mod(\Sigma') \to Mod(\Sigma)$,
- $\models: |\mathbf{Sign}| \to |\mathrm{Rel}^{\to}|$ a function, associating each signature Σ with a binary relation $\models_\Sigma \subseteq Mod(\Sigma) \times Sen(\Sigma)$, viz. the satisfaction relation between Σ-models and Σ-sentences.

Reduction preserves satisfaction in $\mathcal{I}$ (or $\mathcal{I}$ has the *rps* property, or $\mathcal{I}$ is *rps* for short) iff for all signature morphisms $\sigma \in \mathbf{Sign}(\Sigma, \Sigma')$, all sentences $\phi \in Sen(\Sigma)$ and all models $M' \in Mod(\Sigma')$

$$M' \models_{\Sigma'} Sen(\sigma)(\phi) \quad \Rightarrow \quad Mod(\sigma)(M') \models_\Sigma \phi$$

Expansion preserves satisfaction in $\mathcal{I}$ (or $\mathcal{I}$ has the *eps* property, or $\mathcal{I}$ is *eps* for short) iff for all signature morphisms $\sigma \in \mathbf{Sign}(\Sigma, \Sigma')$, all sentences $\phi \in Sen(\Sigma)$ and all models $M' \in Mod(\Sigma')$

$$Mod(\sigma)(M') \models_\Sigma \phi \quad \Rightarrow \quad M' \models_{\Sigma'} Sen(\sigma)(\phi)$$

$\mathcal{I}$ *preserves satisfaction* (or $\mathcal{I}$ has the *ps* property, or $\mathcal{I}$ is *ps* for short) iff $\mathcal{I}$ is both *rps* and *eps*. $\Box$

Thus an institution is a pre-institution that preserves satisfaction and where model sets and reductions have categorical structure.

The focus of [81] is on the *transformations* of pre-institutions and the preservation of logical properties by these morphisms. In particular a technique to prove that the satisfaction condition is satisfied is presented, consisting of coding a pre-institution in an institution (i.e. a pre-institution that is *ps*) preserving satisfaction, so that the satisfactions condition reflects from the codomain to the domain. In the next chapters the precise definition of transformation will be presented and compared with other notions of morphism between institutions.

1.2.2 Specification Logics

Starting from the point of view that the theory of algebraic module specifications and modular systems, although mainly developed in the framework of equational algebraic specifications, is independent from this framework and should be stated in a more general formulation that could be instantiated on any specification formalism, in [33] the core of the theory of algebraic module specification is rephrased in an entirely categorical way, adopting the concept of *specification logic* to represent a generic formalism.

Specification logics are designed to capture a wider range of applications than institutions and intuitively differ from institutions because the satisfaction relation is implicit, with the theory presentations (pairs of signatures and set of sentences, in the institution language) with their categories of models as primitive concept, and validity is not invariant under change of notation. From a technical point of view, specification logics are simply institutions without sentences, i.e. strictly indexed categories.

Def. 1.2.2 [[33], def. 2.1] A *specification logic* $\mathcal{SL}$ is a pair $(\mathbf{ASPEC}, Mod)$ where $\mathbf{ASPEC}$ is a category of *abstract specifications* and $Mod\colon \mathbf{ASPEC} \to \mathbf{Cat}^{Op}$ is a functor that associates with every specification in $\mathbf{ASPEC}$ its category of *models*. □

Note that in the definition of specification logics there is no explicit notion of logic not even of sentence or validity, so that much more than the intended applications are captured.

The relationship between institutions and specification logics is complex. Indeed any institution $\mathcal{I} = (\mathbf{Sign}, Sen, Mod, \models)$ gives rise to a bunch of specification logics, one for each subcategory of its *theory* category.

Def. 1.2.3 Given an institution $\mathcal{I}$ its category $Th(\mathcal{I})$ of *theories* has as objects pairs $T = (\Sigma, \Gamma)$ with Σ a signature and Γ a set of sentences on Σ and as morphism $\sigma\colon (\Sigma, \Gamma) \to (\Sigma', \Gamma')$ the signature morphisms $\sigma\colon \Sigma \to \Sigma'$ s.t. $A' \models_{\Sigma'} \gamma'$ for all $\gamma' \in \Gamma'$ implies $A' \models_{\Sigma'} Sen(\sigma)(\gamma)$ for all $\gamma \in \Gamma$.

The subcategory $\mathbf{Th}_0$ of $Th(\mathcal{I})$ has the same objects as $Th(\mathcal{I})$ and the *axiom-preserving* morphisms $\sigma\colon (\Sigma, \Gamma) \to (\Sigma', \Gamma')$ s.t. $Sen(\sigma)(\Gamma) \subseteq \Gamma'$.

The functor $\mathbf{Mod}_{\mathcal{I}}\colon Th(\mathcal{I}) \to \mathbf{Cat}^{Op}$ associates any theory $T = (\Sigma, \Gamma)$ with the full subcategory $\mathbf{Mod}_{\mathcal{I}}(T)$ of $Mod(\Sigma)$ of the *models* of T, i.e. $|\mathbf{Mod}_{\mathcal{I}}(T)| = \{A \mid A \in |Mod(\Sigma)|, A \models_{\Sigma} \gamma, \gamma \in \Gamma\}$ and any theory morphism $\sigma\colon (\Sigma, \Gamma) \to (\Sigma', \Gamma')$ with the restriction of $Mod(\sigma)$ to $\mathbf{Mod}_{\mathcal{I}}(T')$. □

Note that the definition of morphisms and the satisfaction condition guarantee that if $A' \in |\mathbf{Mod}_{\mathcal{I}}(T')|$, then $Mod(\sigma)(A') \in |\mathbf{Mod}_{\mathcal{I}}(T)|$ for all $\sigma: T \to T'$ and hence $\mathbf{Mod}_{\mathcal{I}}(\sigma)$ is well defined.

Prop. 1.2.4 Let $\mathcal{I}$ be an institution and $\mathbf{C}$ be a (possibly non-full) subcategory of $Th(\mathcal{I})$; then $(\mathbf{C}, \mathbf{Mod}_{\mathcal{I}} \circ Emb)$, where Emb is the embedding of $\mathbf{C}$ into $Th(\mathcal{I})$, is a specification logic.
Proof. Trivial by definition of specification logic. $\square$

In particular any institution induces two specification logics: the trivial one, with signatures as category of abstract specifications, and the "intended" one, with theories as category of abstract specifications. On the other side any specification logic induces a (trivial) institution, with abstract specifications as category of signatures and empty sentence sets (and hence empty satisfaction relations).

Note that if the category of abstract specifications is actually a category of specifications, i.e. there exists a pre-institution $\mathcal{I} = (\mathbf{Sign}, Sen, Mod, \models)$ s.t. $\mathbf{ASPEC} = Th(\mathcal{I})$, then the functoriality of Mod guarantees that $\mathcal{I}$ is *rps*. Thus the intuition behind specification logics is not dropping the satisfaction condition, but weakening the requirement to an implication instead of an equivalence.

Prop. 1.2.5 Let $\mathcal{I}$ be a pre-institution s.t. $(Th(\mathcal{I}), \mathbf{Mod}_{\mathcal{I}})$ is a specification logic; then $\mathcal{I}$ is *rps*.
Proof. Assume that $M' \models_{\Sigma'} Sen(\sigma)(\phi)$ for some signature morphism $\sigma \in \mathbf{Sign}(\Sigma, \Sigma')$, sentence $\phi \in Sen(\Sigma)$ and model $M' \in Mod(\Sigma')$; then, by definition, $M' \in |\mathbf{Mod}_{\mathcal{I}}(\Sigma', \{Sen(\sigma)(\phi)\})|$. Since σ is a theory morphism from $(\Sigma, \{\phi\})$ to $(\Sigma', \{Sen(\sigma)(\phi)\})$ and $\mathbf{Mod}_{\mathcal{I}}$ is a countervariant functor, $\mathbf{Mod}_{\mathcal{I}}(\sigma)(M') \in \mathbf{Mod}_{\mathcal{I}}(\Sigma, \{\phi\})$, i.e. $\mathbf{Mod}_{\mathcal{I}}(\sigma)(M') \models_{\Sigma} \phi$. $\square$

The notions of sentence and validity are reintroduced also in the framework of specification logics, with the satisfaction condition, under the name of *constraints*.

Def. 1.2.6 A *logic of constraints* $\mathcal{LC} = (Constr, \models)$ on a given specification logic $\mathcal{SL} = (\mathbf{ASPEC}, Mod)$ is given by a functor $Constr: \mathbf{ASPEC} \to \mathbf{Classes}$ defined on the category $\mathbf{ASPEC}$ of abstract specifications with values in the (quasi)category $\mathbf{Classes}$ of classes and for each object T in $\mathbf{ASPEC}$ a relation $\models_T \subseteq |Mod(T)| \times Constr(T)$ called *satisfaction relation for constraints*, such that for all morphisms $\sigma: T \to T'$, all objects $A' \in |Mod(T')|$ and all constraints $c \in Constr(T)$ the following *satisfaction condition* holds:

$$A' \models_{T'} Constr(c) \iff Mod(A') \models_T c. \qquad \square$$

Thus any logic of constraints, together with its underline specification logic, correspond to an institution in the sense of [44], exception made for the fact that institutions have *sets* of sentences instead of *classes* of sentences.

1.2.3 Galleries

Mayoh, in [61], suggests a wide range of applications for an institution-like concept that cannot be expressed in the institution formalism, because the notion of validity is too restrictive, including the theory of knowledge representation, data-base query systems and semantics of programming languages.

The definition of *galleries*, broadening the notion of "truth value", captures these examples in a formalism where sentences are substituted with "expressions" and the validity with an "evaluation" relation, so that for instance, given an expression e in a database query language and a database D that is an acceptable model for the language, the evaluation of e in D yields the response to the query e for the database D.

The intuition behind galleries is to relate (by a functor) any language (signature) to a building block, called *room*, consisting of the expressions, called *frames*, the structures on the language and of an evaluation function that on a given expression and a given model produces as output the value of the interpretation of the expression in the model.

Due to increased expressive capability, the notions of specifications (theory) or data type, have to be rephrased and generalized; as in the usual two valued logic a specification (data type) is a set of sentences and its semantics is the class of structures that satisfy the axioms, i.e. where the evaluation of the axioms is the *true* value, here, with a smaller granularity, the semantic of a data type can be defined as the class of structures where the evaluation of the expressions yields some fixed values.

Def. 1.2.7 [[61] def. 3] A *room* consists of a set FRM of *frames*, a category $\mathbf{STR}$ of *structures*, and a functor Val from $\mathbf{STR}$ to $B_n(FRM)$, the category of functions from FRM to $\mathbf{Set}$. The *data types* of the room are the objects in the category $B_n(FRM)$. A data type d is *realisable* iff there exists $m \in |\mathbf{STR}|$ s.t. $d(e) = Val(m)(e)$ for all $e \in FRM$; it is a *truth value* iff $d(e)$ is either the empty or the singleton set for each $e \in FRM$. □

The category $B_n(FRM)$ is the category of functors from FRM, regarded as discrete category, in $\mathbf{Set}$; thus the arrows f in $B_n(FRM)$ from $v: FRM \to \mathbf{Set}$ in $w: FRM \to \mathbf{Set}$ are FRM-indexed families of functions $f_e: v(e) \to w(e)$.

Rephrased in the institution language, a room is (almost) a generalization of the slice of an institution depending on a fixed signature Σ, with FRM as $Sen(\Sigma)$, $\mathbf{STR}$ as $Mod(\Sigma)$ and Val playing the role of a generalized satisfaction relation, because $[\mathbf{STR} \to [FRM \to \mathbf{Set}]]$ is isomorphic to $[\mathbf{STR} \times FRM \to \mathbf{Set}]$. In particular if all the realisable data types are truth values, interpreting the singleton set as the *true* value and the empty set as the *false* value, the two notions seem to coincide (such a room is called *logical*). But note that, following the intuition that Val is an evaluation and that morphisms in any algebraic setting preserve the evaluation of expressions, any model morphism $h\colon A \to B$ corresponds by Val to a family of functions $Val(h)_e\colon Val(A)(e) \to Val(B)(e)$, translating the interpretation of e in A in terms of interpretation of e in B. Thus if both $Val(A)$ and $Val(B)$ are truth values and $Val(A)(e) = true$, then $Val(B)(e) = true$ is forced by the existence of $Val(h)_e\colon Val(A)(e) \to Val(B)(e)$, because there is not a function from $true = \{\bullet\}$ into $false = \emptyset$. Therefore in a logical room only those model morphisms preserving the truth of sentences are allowed, contrary to the more liberal institution formalism. It is worth to note that in the usual algebraic settings homomorphisms do not preserve the truth of sentences, not even in the equational institution; consider indeed the following example.

Example 1.2.8 Let Σ be the signature with one sort, s, a constant symbol, a, and a unary function, f and define the following two Σ-algebras A and B.

Algebra $A =$
$s^A = \{\cdot\}$
$a^A = \cdot$
$f^A(\cdot) = \cdot$

Algebra $B =$
$s^B = \{1, 2\}$
$a^B = 1$
$f^B(x) = x$

Then $h\colon A \to B$, defined by $h(\cdot) = 1$, is obviously a homomorphism; but it is immediate to check that A satisfies the equality $f(x) = a$, while B does not. $\square$

A *gallery* associates each signature with a room, but to deal with changes of notation, morphisms between rooms have to be defined first.

Def. 1.2.9 A *morphism* ϕ from a room $R = (FRM, \mathbf{STR}, Val)$ to a room $R' = (FRM', \mathbf{STR}', Val')$ consists of a function $\phi^6\colon FRM \to FRM'$ and a functor $\phi^\sharp$ from $\mathbf{STR}'$ to $\mathbf{STR}$ such that

$$e \in Val(\phi^\sharp(m')) \iff \phi^6(e) \in Val'(m')$$

for all sentences $e \in FRM$ and structures $m' \in |\mathbf{STR}'|$.

A *gallery* G is a functor to the category of rooms from a category **Sign** of signatures. □

In [61] a gallery yielding a logical room on every signature is called an institution and indeed each such a gallery satisfies the definition of institution in [44]; but note that not every institution in the sense of [44] is a gallery yielding a logical room, because in the gallery formalism the model homomorphisms are restricted to the ones preserving the truth of sentences.

Following the main stream in [61], in [45] the definition of *generalized* institution is given, that introduce in the context of institutions the possibility to have evaluation more than satisfaction, but keep the idea that model homomorphisms are not required to preserve truth. The generalization is easy from a technical point of view, adopting the definition of institution in terms of extranatural transformations.

Def. 1.2.10 [[45] def. 13[4]] Let $\mathbf{V}$ be a category. Then a (*generalized*) $\mathbf{V}$-institution is a pair of functors $Mod\colon \mathbf{Sign}^{op} \to \mathbf{Cat}$ and $Sen\colon \mathbf{Sign} \to \mathbf{Set}$ with an extranatural transformation $\models\colon |Mod(\_)| \times Sen(\_) \to \mathbf{V}$, where the functor $|\_|$ is the forgetful functor from **Cat** to **Set**, discarding the arrows. □

From the above definition the concepts of room and room morphism can be deduced as non-primitive.

Def. 1.2.11 [[45] def. 14] A *generalized* $\mathbf{V}$*-room* consists of categories $\mathbf{M}$ and $\mathbf{S}$ and a functor $r\colon |\mathbf{M}| \to [\mathbf{S} \to \mathbf{V}]$, where $\mathbf{V}$ is a *value* category, $\mathbf{M}$ is a *model* category, $\mathbf{S}$ is a *sentence* category, and $[\mathbf{S} \to \mathbf{V}]$ denotes the functor category.

Let r and r' be generalized $\mathbf{V}$-rooms. The a *generalized* $\mathbf{V}$*-room morphism* from r to r' is a pair of functors $f\colon \mathbf{M}' \to \mathbf{M}$, $g\colon \mathbf{S} \to \mathbf{S}'$ such that the following diagram commutes:

$$\begin{tikzcd}
|\mathbf{M}| \arrow[r, "r"] & {[\mathbf{S} \to \mathbf{V}]} \\
|\mathbf{M}'| \arrow[u, "|f|"] \arrow[r, "r'"'] & {[\mathbf{S}' \to \mathbf{V}]} \arrow[u, "{[g \to \mathbf{V}]}"']
\end{tikzcd}$$

[4] In the original definition the codomain of the sentence functor is **Cat**, and the arrows between sentences play the role of proofs.

where $[g \to \mathbf{V}]$ on an element $\phi\colon \mathbf{S}' \to \mathbf{V}$ yields $\phi \circ g$.

Let $\mathbf{Room}(\mathbf{V})$ denote the category of generalized $\mathbf{V}$-rooms and generalized $\mathbf{V}$-room morphisms; a *generalized institution* is a functor $\mathcal{I}\colon \mathbf{Sign} \to \mathbf{Room}(\mathbf{V})$. $\square$

It is straightforward to verify that the two definitions of generalized institution coincide.

1.2.4 Foundations

The starting point of this approach (see e.g. [75]) in Poigné's words is the *firm belief that any notion of logical system should come along with some notion of "terms" (or rather "operators"), "formulas" and "*substitution*"*.

Following the formalization of predicate logic by indexed categories (see e.g. [52, 53, 87]), the idea is to abstract generalized institutions (see [45]) by requiring that the *sentences* form a fibration instead of a (plain) category.

Def. 1.2.12 A $\mathbf{T}$-*indexed category* consists of

- a category $\mathbf{T}$ (of *derived operators*);
- a category $\mathbf{P}(A)$ (of *properties* or *predicates*) for each object A of $\mathbf{T}$;
- a functor $f^*\colon \mathbf{P}(B) \to \mathbf{P}(A)$ for each arrow $f\colon A \to B$ in $\mathbf{T}$ such that
  - $Id_A^*\colon \mathbf{P}(A) \to \mathbf{P}(A)$ is naturally isomorphic to the identity functor;
  - for all arrows $f\colon B \to C$ and $g\colon A \to B$ in $\mathbf{T}$ the functor $(f \circ g)^*$ is naturally isomorphic to $f^* \circ g^*$

  such functors are called *(predicate) transformers.* $\square$

In the case of predicate logic, the category $\mathbf{T}$ has sequences of sorts as objects and k-tuples of terms of sort s'_j on variables x_i of sort s_i, with $i = 1 \ldots n$, as arrows from $s_1, \ldots, s_n$ into $s'_1, \ldots, s'_k$; composition is term substitution. For any sequence $A = s_1, \ldots, s_n$ of sorts, $\mathbf{P}(A)$ has formulas on variables x_i of sort s_i, with $i = 1 \ldots n$, as objects and proofs as arrows; finally the functor associated with a tuple $t_1, \ldots, t_n$ of terms on variables x_i of sort s_i, with $i = 1 \ldots k$, is the substitution of the terms for the variables in formulas.

An equivalent definition of indexed categories that seems to be more convenient in the present context is that of *fibration*.

Def. 1.2.13 Let $p: \mathbf{P} \to \mathbf{T}$ be a functor, X an object of $\mathbf{P}$ and $\alpha: A \to p(X)$ an arrow of $\mathbf{T}$. A morphism $f: \alpha^* X \to X$ is *horizontal* over α iff $p(f) = \alpha$ and any arrow in $\mathbf{P}$ whose image along p factorizes through α factorizes through f, i.e. for any $g: Y \to X$ s.t. $p(g) = \alpha \circ \beta$ there exists $h: Y \to \alpha^* X$ s.t. $p(h) = \beta$ and $g = f \circ h$.

A functor $p: \mathbf{P} \to \mathbf{T}$ is called a *fibration* if given any object X of $\mathbf{P}$ and any arrow $\alpha: A \to p(X)$ of $\mathbf{T}$ there is some horizontal $f: \alpha^* X \to X$ over α. Then $\mathbf{T}$ is called the *base* of a fibration and $\mathbf{P}$ the *plateau*.

A morphism of fibrations from $p: \mathbf{P} \to \mathbf{T}$ into $p': \mathbf{P}' \to \mathbf{T}'$ consists of two functors $F_T: \mathbf{T} \to \mathbf{T}'$ and $F_P: \mathbf{P} \to \mathbf{P}'$ s.t. $p'(F_P(X)) = F_T(p(X))$ for any object X of $\mathbf{P}$ and $(F_T(\alpha))^* F_P(X) = F_P(\alpha^* X)$ for any arrow $\alpha: A \to p(X)$ of $\mathbf{T}$.

A fibration is called *small* if all the categories involved are small. □

$\mathbf{T}$-indexed categories are fibrations in the sense that for any $\mathbf{T}$-indexed category a fibration is obtained by constructing $\mathbf{P}$ with objects $< A, X >$, A being an object of $\mathbf{T}$ and X being an object of $\mathbf{P}(A)$. Morphisms are pairs $< \alpha, f >: < A, X > \to < B, Y >$, with $\alpha: A \to B$ and $f: X \to \alpha^* Y$.

Thus on one hand the concept of *foundation* is more concrete than the one of institutions, for sentences are in some sense specialized, and on the other is more general, because tools to deal with proofs are introduced. Following the gallery approach by Mayoh, sentences are evaluated more than satisfied.

Def. 1.2.14 A *rich institution* consists of functors, $\mathit{Frame}: \mathbf{Sign} \to \mathbf{Cat} \downarrow \mathbf{Cat}$ and $\mathit{Mod}: \mathbf{Sign}^{Op} \to \mathbf{Cat}$, and a wedge $[\_]: |\mathit{Mod}(\_)| \times \mathit{Frame}(\_) \to \mathbf{V}$ (in $\mathbf{Cat} \downarrow \mathbf{Cat}$).

A *foundation* is a rich institution such that

- $\mathit{Frame}(\Sigma)$ factorizes over the category of small fibrations,
- $\mathbf{V}$ is a fibration, and
- $[\_]_{\Sigma,C}: \mathit{Frame}(\Sigma) \to \mathbf{V}$ is a morphism of fibrations for every C in $\mathit{Mod}(\Sigma)$. □

Since any category is an indexed category too (in the trivial way), any institution is a foundation, too. Note, however, that this trivial lifting of institutions to foundations does not correspond to the intuition of typed sentences as frames behind the introduction of indexed categories in the case of ranked signatures.

In [76] this approach is made more complex and expressive, by the introduction of deduction system to specialize the nature of frames. The formal complexity of the categorical theory involved makes doubtfully whether the heavy formalism is worthwhile, in front of the tools provided to handle proofs and sentences.

1.2.5 π-Institutions and Entailment Systems

Two approaches sharing not only the intuition of the entailment (consequence) relation as the primitive and central concept, but also most technical points are the *π-institutions* by Fiadeiro and Sernadas (see e.g. [36]) and the *entailment systems* by Meseguer (see e.g. [63]).

The basic idea is to define a logical system through a primitive notion of *consequence*, moving the focus away from the models to the deduction; this approach seems greatly promising, especially considering the widening use of mechanical theorem provers as supporting tools for the specification design.

The π-institutions are proposed as an alternative to institutions, replacing the notion of model and satisfaction by a primitive consequence operator (à la Tarski); then in [36] it is shown how theories can be manipulated in the framework of π-institutions towards the desired semantics of specification building.

Instead the notion of entailment system is a part of a bigger integrated framework, where tools to represent entailment, proofs and calculi are at hand and institutions are the fragment of the fresco dealing with the semantic side.

Although the two works follow different development lines, the only difference between the definitions of π-institution and entailment system consists of the assumption of compactness for the π-institutions; thus the two theories are summarized together.

The following definition of π-institution generalizes the notion of *deductive system* in the sense of [94], relativizing the primitive concepts of *proposition* and *consequence* subject to a given signature and then characterizing the behaviour of these concepts under changes of notation.

Def. 1.2.15 [[36], def. 2.1] A *π-institution* is a triple $(\mathbf{Sign}, Sen, \{\mathbf{Cn}_\Sigma\}_{\Sigma\in|\mathbf{Sign}|})$ consisting of

1. A category $\mathbf{Sign}$ (of *signatures*);
2. a functor $Sen: \mathbf{Sign} \to \mathbf{Set}$ (giving the set of formulas over each signature);
3. for each object $\Sigma \in |\mathbf{Sign}|$, a *consequence operator* $\mathbf{Cn}_\Sigma$ defined in the power set of $Sen(\Sigma)$ satisfying for each $\Gamma, \Delta \subseteq Sen(\Sigma)$ and $\sigma: \Sigma \to \Sigma'$:

   extensiveness $\Gamma \subseteq \mathbf{Cn}_\Sigma(\Gamma)$;

   idempotence $\mathbf{Cn}_\Sigma(\mathbf{Cn}_\Sigma(\Gamma)) = \mathbf{Cn}_\Sigma(\Gamma)$;

   compactness $\mathbf{Cn}_\Sigma(\Gamma) = \bigcup_{\Delta\subseteq\Gamma,\Delta \text{finite}} \mathbf{Cn}_\Sigma(\Delta)$;

   structurality $Sen(\sigma)(\mathbf{Cn}_\Sigma(\Gamma)) \subseteq \mathbf{Cn}_{\Sigma'}(Sen(\sigma)(\Gamma))$. □

Def. 1.2.16 [[63], def. 1] An *entailment system* is a triple

$$\mathcal{E} = (\mathbf{Sign}, Sen, \vdash)$$

with **Sign** a category whose objects are called *signatures*, Sen a functor $Sen\colon \mathbf{Sign} \to \mathbf{Set}$ and $\vdash$ a function, associating with each Σ in **Sign** a binary relation $\vdash_\Sigma \subseteq \wp(Sen(\Sigma)) \times Sen(\Sigma)$ called Σ-*entailment* such that the following properties are satisfied:

reflexivity for any $\phi \in Sen(\Sigma)$, $\{\phi\}\vdash_\Sigma\phi$;

monotonicity if $\Gamma\vdash_\Sigma\phi$ and $\Gamma \subseteq \Gamma'$, then $\Gamma'\vdash_\Sigma\phi$;

transitivity if $\Gamma\vdash_\Sigma\phi_i$ for $i \in I$ and $\Gamma \cup \{\phi_i \mid i \in I\}\vdash_\Sigma\psi$, then $\Gamma\vdash_\Sigma\psi$;

$\vdash$-*translation* if $\Gamma\vdash_\Sigma\phi$, then for any $\sigma \in \mathbf{Sign}(\Sigma, \Sigma')$, $Sen(\sigma)(\Gamma) \vdash_{\Sigma'} Sen(\sigma)(\phi)$.

The *closure* $\{\phi \mid \Gamma \vdash \phi\}$ of a set Γ is denoted by $\Gamma^\bullet$.

An entailment system is called *compact* iff for each $\Sigma \in |\mathbf{Sign}|$ and each $\Gamma \subseteq Sen(\Sigma)$

$$\Gamma\vdash_\Sigma\phi \iff \text{there exists finite } \Delta \subseteq \Gamma s.t.\ \Delta\vdash_\Sigma\phi$$

It is just an exercise to show that compact entailment systems and π-institutions coincide.

Prop. 1.2.17 Let $\mathcal{E} = (\mathbf{Sign}, Sen, \vdash)$ be an entailment system and $\Pi = (\mathbf{Sign}, Sen, \{\mathbf{Cn}_\Sigma\}_{\Sigma\in|\mathbf{Sign}|})$ be a π-institution.

1. $F(\Pi) = (\mathbf{Sign}, Sen, \vdash^{\mathbf{Cn}})$ is a compact entailment system, where $\vdash^{\mathbf{Cn}} = \{\vdash^{\mathbf{Cn}}{}_\Sigma\}_{\Sigma\in|\mathbf{Sign}|}$ is defined by: $\Gamma\vdash^{\mathbf{Cn}}{}_\Sigma\phi$ iff $\phi \in \mathbf{Cn}_\Sigma(\Gamma)$ for all $\Gamma \subseteq Sen(\Sigma)$ and $\phi \in Sen(\Sigma)$.

2. $G(\mathcal{E}) = (\mathbf{Sign}, Sen, \mathbf{Cn}^\vdash)$ is a π-institution, where $\mathbf{Cn}^\vdash = \{\mathbf{Cn}^\vdash{}_\Sigma\}_{\Sigma\in|\mathbf{Sign}|}$ is defined by: $\mathbf{Cn}^\vdash(\Gamma) = \bigcup_{\Delta\subseteq\Gamma, \Delta \text{ finite}} \Delta^\bullet$

3. $G(F(\Pi)) = \Pi$;

4. if $\mathcal{E}$ is compact, then $F(G(\mathcal{E})) = \mathcal{E}$. $\square$

Proof.

1. Extensiveness immediately implies reflexivity, compactness implies monotonicity and $\vdash$-translation directly follows from structurality. To check transitivity, assume that $\Gamma \vdash^{\mathbf{Cn}}{}_\Sigma \phi_i$ for $i \in I$ and $\Gamma \cup \{\phi_i \mid i \in I\} \vdash^{\mathbf{Cn}}{}_\Sigma \psi$, i.e. that $\{\phi_i \mid i \in I\} \subseteq \mathbf{Cn}_\Sigma(\Gamma)$ and $\psi \in \mathbf{Cn}_\Sigma(\Gamma \cup \{\phi_i \mid i \in I\})$; then, by extensiveness, $\Gamma \cup \{\phi_i \mid i \in I\} \subseteq \mathbf{Cn}_\Sigma(\Gamma)$ and hence, by monotonicity, $\psi \in \mathbf{Cn}_\Sigma(\mathbf{Cn}_\Sigma(\Gamma))$ so that, by idempotence, $\psi \in \mathbf{Cn}_\Sigma(\Gamma)$, i.e. transitivity holds, too.

2. Reflexivity and monotonicity immediately imply extensiveness.

   From extensiveness $\mathbf{Cn}^\vdash{}_\Sigma(\Gamma) \subseteq \mathbf{Cn}^\vdash{}_\Sigma(\mathbf{Cn}^\vdash{}_\Sigma(\Gamma))$; thus to show that idempotence holds, it is sufficient to show that $\mathbf{Cn}^\vdash{}_\Sigma(\mathbf{Cn}^\vdash{}_\Sigma(\Gamma)) \subseteq \mathbf{Cn}^\vdash{}_\Sigma(\Gamma)$. By definition
   $$\mathbf{Cn}^\vdash{}_\Sigma(\mathbf{Cn}^\vdash{}_\Sigma(\Gamma)) = \bigcup_{\Delta \subseteq [\cup_{\Theta \subseteq \Gamma, \Theta \text{ finite}} \Theta^\bullet], \Delta \text{ finite}} \Delta^\bullet.$$
   But $\Delta \subseteq (\bigcup_{\Theta \subseteq \Gamma, \Theta \text{ finite}} \Theta^\bullet)$ and Δ finite, i.e. $\Delta = \{\delta_i \mid i = 1 \ldots n\}$, imply $\Delta^\bullet \subseteq (\bigcup_{\Theta \subseteq \Gamma, \Theta \text{ finite}} \Theta^\bullet)$, because for each δ_i there exists a finite Θ_i s.t. $\delta_i \in \Theta_i{}^\bullet$ for $i = 1 \ldots n$, so that $\Delta \subseteq (\bigcup_{i=1\ldots n} \Theta_i{}^\bullet) \subseteq \Theta^\bullet$ and $\Theta = \bigcup_{i=1\ldots n} \Theta_i$ is finite. Therefore
   $$\bigcup_{\Delta \subseteq [\cup_{\Theta \subseteq \Gamma, \Theta \text{ finite}} \Theta^\bullet], \Delta \text{ finite}} \Delta^\bullet \subseteq \bigcup_{\Theta \subseteq \Gamma, \Theta \text{ finite}} \Theta^\bullet$$
   i.e. $\mathbf{Cn}^\vdash{}_\Sigma(\mathbf{Cn}^\vdash{}_\Sigma(\Gamma)) \subseteq \mathbf{Cn}^\vdash{}_\Sigma(\Gamma)$.

   Compactness follows from the obvious fact that if Γ is finite, then $\mathbf{Cn}^\vdash{}_\Sigma(\Gamma) = \Gamma^\bullet$.

   Finally structurality easily follows from $\vdash$-translation.

3. Obvious, because of compactness.

4. By definition, because of compactness, $\mathbf{Cn}^\vdash(\Gamma) = \Gamma^\bullet$. □

In both framework a concept of model is provided for the theories, using the deductive relation, in order to make institutions and both compare the different approaches and import the results.

In the case of entailment systems the construction of the model part is easier and obviously applies to the π-institutions too, because π-institutions are entailment systems.

Def. 1.2.18 Given an entailment system [π-institution] its category **Th** of *theory presentations* has as objects pairs $T = (\Sigma, \Gamma)$ with Σ a signature and Γ a set of sentences on Σ and as morphism $\sigma: (\Sigma, \Gamma) \to (\Sigma', \Gamma')$ the signature morphisms $\sigma: \Sigma \to \Sigma'$ s.t. $Sen(\sigma)(\Gamma) \subseteq \Gamma'^{\bullet}$ $[Sen(\sigma)(\Gamma) \subseteq \mathbf{Cn}_{\Sigma'}(\Gamma')]$.

The full subcategory **The** of **Th** has closed theory presentations as objects, i.e. theory presentations (Σ, Γ) s.t. $\Gamma = \Gamma^{\bullet}$ [resp. $\Gamma = \mathbf{Cn}_{\Sigma}(\Gamma)$]. □

Note that, using the notation of Prop. 1.2.17, the definition of theory category for any π-institution Π coincides with the definition of theory category for the entailment system $F(\Pi)$.

Prop. 1.2.19 Let $\mathcal{E} = (\mathbf{Sign}, Sen, \vdash)$ be an entailment system; then $\mathcal{I} = (\mathbf{Sign}, Sen, Mod, \models)$ is an institution, where $Mod(\Sigma)$ is the comma category $(\Sigma, \emptyset) \downarrow \mathbf{Th}$ (see e.g. [54]) and $Mod(\sigma)$ is the right composition $Mod(\sigma)(\tau) = \tau \circ \sigma$ on the objects and leaves the arrows unaffected.
Proof. See prop. 9 in [63]. □

Vice-versa any institution implicitly defines two entailment systems: the minimal one, where a set of sentences entails just its elements, and the maximal (or complete) one, where a set of sentences entails the sentences that hold in its models. Lifting up the discussion to a categorical level, in [63], the relationships between entailment systems and institutions is clearly illustrated by a graph, whose edges are (forgetful and their left/right adjoint) functors and whose nodes are the categories of institutions, of entailment systems and of *logics* (institutions endowed with a sound entailment system). This discussion is referred on in chapter 4.

In [36] a more complex construction is proposed, that follows largely the same path, but starting with a subcategory of arrows in **The** as models. The intuition is that a theory morphism $\sigma: T \to T'$ is an *interpreter* of T in T' (note that the category of interpreters of T coincides with the comma category $T \downarrow T'$), but only the fragments of an interpretation that are image of the interpreter are relevant to the interpretation itself. These parts, indeed, are the *nucleus* of the interpretation and contain all (and only) the information characterizing the interpretation. In this way many interpretations are identified, dropping the inessential parts. Two properties appear to characterize the notion of nucleus: on one hand the nucleus, being the "essential" part of an interpreter, should be in some sense minimal, and on the other hand any interpreter should be a conservative extension of its nucleus, i.e., roughly speaking, the nucleus "plus" a disjoint (inessential) part. In a set theoretic approach these requirements would be satisfied, factorizing $\sigma: T \to T'$ in the composition of σ itself, viewed now as a function from T into the image T^* of σ, with the embedding of T^* into T' and then using as nucleus $\sigma: T \to T^*$, minimal in the sense of surjective, so that $\sigma: T \to T'$ is a conservative extension of the nucleus in the sense that $\sigma = \iota \circ \sigma$ and the embedding ι is injective. This approach can be generalized, provided that the category of theories has an image factorization system, playing the role of the surjective-injective factorization for the set theoretic case.

Def. 1.2.20 [[36], def. 3.8] Given a category $\mathbf{C}$, an *image factorization system for* $\mathbf{C}$ is a pair (E, M) such that:

- E is a class of epimorphisms in $\mathbf{C}$ and M is a class of monomorphisms in $\mathbf{C}$.
- E and M are closed under composition.
- E and M contain all isomorphisms in $\mathbf{C}$.
- Every morphism f in $\mathbf{C}$ admits an (E, M)-factorization that is unique up to isomorphism. That is to say, there exist $e \in E$ and $m \in M$ such that $f = m \circ e$ and if f admits another (E, M)-factorization $f = m' \circ e'$, then there is an isomorphism h such that $h \circ e = e'$ and $m' \circ h = m$. $\square$

The existence of an image factorization system for the category of theories can be reduced to the existence of an image factorization system for the category of signatures, so that just the syntactic part is involved.

Theorem 1.2.21 Let (E, M) be an image factorization system for **Sign**. Let ET be the class of all theory morphisms σ belonging to E. Let MT be the class of all conservative theory morphisms σ belonging to M, where a theory morphism $\sigma: (\Sigma, \Gamma) \to (\Sigma', \Gamma')$ is called *conservative* iff $\sigma^{-1}(\Gamma') \subseteq \Gamma$.

Then (ET, MT) is an image factorization system for **The**.

Proof. See [36], theorem 3.10. $\square$

In the sequel the category **Sign** of signatures is assumed to have an image factorization system (E, M).

Def. 1.2.22 [[36], def. 3.12, 3.14, 3.33] Given a theory T in **The**, a *T-interpretation* is a theory T' together with a morphism $\sigma: T \to T'$, called *interpreter*.

For each theory T the category $\iota(T)$ of T-interpretations is the comma category $T \downarrow \mathbf{The}$.

Given an interpreter $\sigma: T \to T'$, the *nucleus* of (σ, T) is the interpretation (e, T^*), where T^* is the center theory for an (ET, MT) factorization of σ with epi e.

Given a signature Σ, the category $Mod(\Sigma)$ of Σ-models is the full subcategory of the comma category $\iota(\Sigma, \mathbf{Cn}_\Sigma(\emptyset))$ whose objects are epimorphisms. □

In order to define how Σ-models are translated along signature morphisms, a preliminary lemma is needed.

Lemma 1.2.23 Let (E, M) be an image factorization system for a category $\mathbf{C}$. Given a commutative square $g \circ e = m \circ f$ with $e \in E$ and $m \in M$, then there is a unique h such that $h \circ e = f$ and $m \circ h = g$.
Proof. See [36], lemma 3.9. □

Note that, using the above lemma, a functor from $\iota(\bar{T})$ to its full subcategory $\rho(\bar{T})$ of epimorphisms can be defined, associating each interpreter with its nucleus and each arrow $\mu: \sigma \to \sigma'$, i.e. $\mu: T \to T'$ s.t. $\mu \circ \sigma = \sigma'$, with the unique $h: T^* \to T'^*$ existing because of the above lemma, for $f = e'$ and $g = \mu \circ m$; graphically:

$$\begin{tikzcd}
T^* \arrow[rr, "h"] \arrow[dd, "m"'] & & T'^* \arrow[dd, "m'"] \\
& \bar{T} \arrow[ul, "e"'] \arrow[ur, "e'"] \arrow[dl, "\sigma"] \arrow[dr, "\sigma'"'] & \\
T \arrow[rr, "\mu"'] & & T'
\end{tikzcd}$$

Def. 1.2.24 [[36], def. 3.33] Let $\bar{T}$ and $\bar{T}'$ denote $(\Sigma, \mathbf{Cn}_\Sigma(\emptyset))$ and $(\Sigma', \mathbf{Cn}_{\Sigma'}(\emptyset))$ respectively. For any signature morphism $\mu: \Sigma \to \Sigma'$ the functor $Mod(\mu): Mod(\Sigma') \to Mod(\Sigma)$ is defined by:

- for any Σ'-model $\sigma': \bar{T}' \to T'$ the image $Mod(\mu)(\sigma')$ is the nucleus of $\sigma' \circ \mu$;
- let $\sigma_1: \bar{T}' \to T'_1$ and $\sigma_2: \bar{T}' \to T'_2$ be Σ'-models and $\phi: \sigma_1 \to \sigma_2$ be a morphism; let (e_1, m_1) and (e_2, m_2) denote the (ET, MT)-factorization of $\sigma_1 \circ \mu$ and $\sigma_2 \circ \mu$ respectively. Then the image $Mod(\mu)(\phi)$ is the unique arrow that make the following diagram commute (that exists because of lemma 1.2.23).

$$
\begin{tikzcd}
{T_1}^* \arrow[rrr, "m_1"] \arrow[ddd, "Mod(\mu)(\phi)"'] & & & T_1' \arrow[ddd, "\phi"] \\
& & & \\
& \bar{T} \arrow[r, "\mu"] \arrow[uul, "e_1"'] \arrow[dl, "e_2"] & \bar{T}' \arrow[uur, "\sigma_1"] \arrow[dr, "\sigma_2"'] & \\
{T_2}^* \arrow[rrr, "m_2"'] & & & T_2'
\end{tikzcd}
$$

Theorem 1.2.25 Let $\Pi = (\mathbf{Sign}, Sen, \{\mathbf{Cn}_\Sigma\}_{\Sigma\in|\mathbf{Sign}|})$ be a π-institution such that **Sign** admits an image factorization system (ET, MT). Then $\mathcal{I} = (\mathbf{Sign}, Sen, Mod, \models)$ is an institution, where *Mod* is defined as in Def. 1.2.22 and Def. 1.2.24 and the relation $\models$ is defined by: $\sigma \models_\Sigma w$ iff $Sen(\sigma)(w) \in \mathbf{Cn}_{\Sigma'}(\Gamma')$ for all $\sigma\colon (\Sigma, \mathbf{Cn}_\Sigma(\emptyset)) \to (\Sigma', \Gamma')$.

Proof. See [36], theorem 3.36. $\square$

Chapter 2

A paradigmatic problem: the Specification of Partial Functions

> The need for a systematic treatment of partial operations is clear from practice. One must be able to handle *errors* and *exceptions*, and account for *non-terminating operations*. There are several approaches to deal with these in literature, none of which appears to be fully satisfactory. *S. Feferman* [35]

Because of the great relevance of the problem and the proliferation of formalisms proposed to deal with it, the specification of partial functions can be assumed as a paradigmatic example of the need for comparing and relating solutions expressed in different frameworks.

Partial operations, besides being a useful tool to represent not yet completely specified functions during the design refinement process, are needed to represent recursive functions. In the practice partiality arises from situations that can be roughly parted in three categories:

- a semidecidable predicate p has to be specified, like in concurrency theory the *transition* relation on processes, or the *typing* relation for higher-order languages. Thus, representing p as a boolean function f_p, it is possible to (recursively) axiomatize the truth, but not whether f_p yields *false* on some inputs and hence f_p is partial (or its image is larger than the usual boolean values set);

- a usual total abstract data type, like the positive natural numbers, is enriched by a partial function, like the subtraction; most of the examples take place in this category, like the famous case of the *stacks*, where the stacks are built by the total functions *empty* and *push* and then *pop* and *top* are defined on them (i.e. the result of the application of these operations is either an "error" or a term on the primitive operations);

- the partial functions that have to be specified are the "constructors" of their image set; consider for example the definition of *lists* without repetitions of elements, or of *ordered trees*; in both cases the new data type is constructed by partial inserting operations.

Although this characterization of partiality cannot be formalized, depending on which kind of problems is considered as the paradigmatic example of partiality, the different frameworks proposed to specify partial functions are more effective on some cases than on others.

The following parade of different formalisms to deal with partiality is slightly reminiscent of [95] and the discussion in [35]. For an analysis of the relationships between different total approaches to the specification of the *stacks* as paradigmatic partial data type, see [17], the first paper that addressed the problem of the proliferation of formalisms from the point of view of the representation of concrete specifications.

2.1 The Total Approaches

In this section the approaches are grouped, where only total functions are used and elements are in a way or the other labeled as "good values" or "errors" of some kind. Note that these approaches also support the treatment of (some kind of) error recovery.

2.1.1 Error Algebras

Error algebras, as introduced by the ADJ group in [41], are a special case of many-sorted algebras; the intuitive idea is to add a constant symbol for each sort representing the "errors" of this sort (and hence an element to each carrier of the data type), carefully propagating the errors. The naive application of this technique can produce a lot of troubles, as the following famous example shows.

Example 2.1.1 Let the natural numbers (positive integers) with the $+$ and the $*$ operations be considered.

spec *Nat* $=$
  sorts N

opns

$zero: \to N$
$S: N \to N$
$+: N \times N \to N$
$*: N \times N \to N$

axioms

$\alpha_1 \quad zero + x = x$
$\alpha_2 \quad S(x) + y = S(x + y)$
$\alpha_3 \quad zero * x = zero$
$\alpha_4 \quad S(x) * y = (x * y) + y$

In order to enrich *Nat* by the $-$ operation an element to represent the error obtained computing $x - y$ for y greater than x has to be provided.[1]

It is worth to stress that in this approach there are operations that, more to be partial strictly speaking, can produce a (predictable) error; thus design refinement and non-recursive function specification are not supported.

spec $Nat^* =$
enrich *Nat* **by**
opns

$err: \to N$
$-: N \times N \to N$

axioms

$\alpha_5 \quad x - zero = x$
$\alpha_6 \quad S(x) - S(y) = x - y$

$\beta \quad zero - S(x) = err$

$\beta_S \quad S(err) = err$
${\beta_+}^l \quad err + x = err$
${\beta_+}^r \quad x + err = err$
${\beta_*}^l \quad err * x = err$
${\beta_*}^r \quad x * err = err$
${\beta_-}^l \quad err - x = err$
${\beta_-}^r \quad x - err = err$

The β axiom introduces the error to denote the subtraction of a positive number from 0 and the decorated β axioms propagate the error.

[1]See [74] for an argumentation against the introduction of error elements in basic types by the hierarchic building of more complex specifications

It is immediate to see that from α_3 and $\beta_*{}^r$ the equality $zero = err$ follows, so that by error propagation any term is equal to err.

To avoid this problem, the axioms $\alpha_1 \ldots \alpha_6$[2] should apply only to the correct elements, i.e. to elements that are different from err. Since inequalities are not allowed by the standard algebraic theory, to express that the variables only range on element distinct from err, a boolean sort is introduced together with an ok function, that is false on err and true on the other elements.

spec $Nat^\star$ =
 sorts N, B
 opns
 $zero\colon \to N$
 $S\colon N \to N$
 $+\colon N \times N \to N$
 $*\colon N \times N \to N$
 $-\colon N \times N \to N$
 $T\colon \to B$
 $F\colon \to B$
 $\wedge\colon B \times B \to B$
 $if_N\colon B \times N \times N \to N$
 $err\colon \to N$
 $ok\colon N \to B$
 $if\_ok_1\colon N \times N \to N$
 $if\_ok_2\colon N \times N \times N \to N$
 axioms
 α'_1 $if\_ok_1(x, zero + x) = if\_ok_1(x, x)$
 α'_2 $if\_ok_2(x, y, S(x) + y) = if\_ok_2(x, y, S(x + y))$
 α'_3 $if\_ok_1(x, zero * x) = if\_ok_1(x, zero)$
 α'_4 $if\_ok_2(x, y, S(x) * y) = if\_ok_2(x, y, (x * y) + y)$
 α'_5 $if\_ok_1(x, x - zero) = if\_ok_1(x, x)$
 α'_6 $if\_ok_2(x, y, S(x) - S(y)) = if\_ok_2(x, y, x - y)$

 α'_7 $T \wedge x = x$
 α'_8 $F \wedge x = F$
 α'_9 $if_N(T, x, y) = x$
 α'_{10} $if_N(F, x, y) = y$

 β_1 $if\_ok_1(x, u) = if_N(ok(x), u, err)$
 β_2 $if\_ok_2(x, y, u) = if_N(ok(x) \wedge ok(y), u, err)$

[2]the only problematic axiom is α_3, but the treatment is wanted to be as uniform as possible

$\beta_3 \quad ok(err) = F$
$\beta_4 \quad ok(zero) = T$
$\beta_5 \quad ok(S(x)) = ok(x)$

$\beta \quad zero - S(x) = err$
$\beta_S \quad S(err) = err$
${\beta_+}^l \quad err + x = err$
${\beta_+}^r \quad x + err = err$
${\beta_*}^l \quad err * x = err$
${\beta_*}^r \quad x * err = err$
${\beta_-}^l \quad err - x = err$
${\beta_-}^r \quad x - err = err$

The above approach can be generalized to the specification of a generic data type, with an *error*, an *ok* and an *if_then_else* for each sort and a $S^* \times S$-family of $\mathit{If\_ok}_{s_1,\ldots,s_n,s}$ functions mapping terms on variables $x_i : s_i$ for $i = 1 \ldots n$ to *error* if any x_i is instantiated on *error* (so that substituting each original axiom $u = v$ of the specification on variables $x_i : s_i$ for $i = 1 \ldots n$ with $\mathit{If\_ok}_{s_1,\ldots,s_n,s}(u) = \mathit{If\_ok}_{s_1,\ldots,s_n,s}(v)$, the new formulation of the axiom is trivially satisfied for the evaluations of variables on error elements). Thus non-empty error signatures must be infinite, having an $\mathit{If\_ok}_{s_1,\ldots,s_n,s}$ for each $n \in \mathbb{N}$ and $s_i \in S$.

Def. 2.1.2 The institution of *error* algebras with (open) equational formulas is the quadruple $\mathcal{ERR} = (\mathbf{Sign}_{\mathcal{ERR}}, Sen_{\mathcal{ERR}}, Mod_{\mathcal{ERR}}, \models_{\mathcal{ERR}})$, where:

- $\mathbf{Sign}_{\mathcal{ERR}}$ is the subcategory of $\mathbf{Sign}_{\mathcal{MS}}$ whose objects (S, F) include the following signature $\Sigma_{bool}(S)$ and whose morphisms preserve the boolean sort and the operation symbols of $\Sigma_{bool}(S)$.

  sig $\Sigma_{bool}(S) =$
  sorts B, s for all $s \in S$
  opns
  $T : \to B$
  $F : \to B$
  $\_ \wedge \_ : B \times B \to B$
  $error_s : \to s$ for all $s \in S \cup \{B\}$
  $ok_s : s \to B$ for all $s \in S \cup \{B\}$
  $ife_s : B \times s \times s \to s$ for all $s \in S$
  $ifok_{w,s} : s_1 \times \ldots \times s_n \times s \to s$ for all $s_1, \ldots, s_n, s \in S$ and $w = s_1 \ldots s_n$

- $Sen_{\mathcal{ERR}} : \mathbf{Sign}_{\mathcal{ERR}} \to \mathbf{Set}$ is the restriction of $Sen_{\mathcal{EMS}}$ to the signatures in $\mathbf{Sign}_{\mathcal{ERR}}$.

- for every signature $\Sigma = (S, F)$ in $\mathbf{Sign}_{\mathcal{ERR}}$, $Mod_{\mathcal{ERR}}(\Sigma) = \mathbf{Mod}_{\mathcal{MS}}(\Sigma, Ax)$, where Ax consists of the following sentences for all $s \in S$:

$$\begin{aligned} \mathit{ife}_s(T, x, y) &= x \\ \mathit{ife}_s(F, x, y) &= y \\ \mathit{ife}_s(\mathit{error}_B, x, y) &= \mathit{error}_s \\ \mathit{ok}_s(\mathit{error}_s) &= F \\ \mathit{ifok}_{s_1,\dots,s_n,s}(x_1, \dots, x_n, u) &= \mathit{ife}_s(\mathit{ok}_{s_1}(x_1) \wedge \dots \wedge \mathit{ok}_{s_n}(x_n), u, \mathit{error}_s) \end{aligned}$$

  and the translation $Mod_{\mathcal{ERR}}(\sigma)$ along a signature morphism $\sigma\colon \Sigma \to \Sigma'$ is the restriction of $Mod_{\mathcal{MS}}(\sigma)$ to $Mod_{\mathcal{ERR}}(\Sigma')$.

- $\models_{\mathcal{ERR}_\Sigma}$ is the restriction of $\models_{\mathcal{MS}}$ to $|Mod_{\mathcal{ERR}}(\Sigma)| \times Sen_{\mathcal{ERR}}(\Sigma)$. □

Note that, since the operation symbols in $\Sigma_{bool}(S)$ are unaffected by signature morphisms in $\mathbf{Sign}_{\mathcal{ERR}}$, the translations of the axioms in Ax along such morphisms are just embeddings and hence for every $\sigma \in \mathbf{Sign}_{\mathcal{ERR}}(\Sigma, \Sigma')$ if an algebra A' is in $Mod_{\mathcal{ERR}}(\Sigma')$, then its reduct $Mod_{\mathcal{MS}}(A')$ is in $Mod_{\mathcal{ERR}}(\Sigma)$, because of the satisfaction condition; thus $Mod_{\mathcal{ERR}}(\sigma)$ is consistently defined.

Note also that there is no reason to assume that just one error element is allowed (although in the original formulation axioms to identify the errors to error_s are in the examples and the construction of canonical error algebras has one error element in each sort); therefore, regarding the elements on which ok_s yields *false* as errors, in this approach tools for as sophisticated as wanted error messages and even error recovery can be found.

It is worth noting that part of the complexity and heaviness of this approach is due to the difficulty of expressing predicates and consequences in the frame of many-sorted algebras (chosen by historical reasons) with equalities as sentences; for example the introduction of the $\mathit{If\_ok}_{s_1,\dots,s_n,s}$ functions is unnecessary if conditional axioms are allowed, with the possibility of adding the sentences $\mathit{ok}_s(x)$ to the premises for all variables x of sort s. Analogously, using first-order structures (algebras with predicates) instead of (plain) many-sorted algebras, the introduction of the boolean sort and related operations can be avoided.

Many other approaches flourished from the original error algebras, refining the basic idea of cataloging the elements of the data type but using more powerful algebraic framework to express the specifications (see e.g. the *exception algebras* in [18], where both the elements of algebras and the terms are labeled to capture the difference between errors and exceptions, or the *clean algebras* in [38], where an order-sorted approach is adopted to catalogue the elements of algebras). In spite of the potentiating and the embellishments, these approaches share with the original one the difficulties of interaction with the modular definition of data types. Indeed the *non-ok* elements of basic types have to be designed *a priori* to support error messages, or exceptions caused by other modules that use the basic ones; thus error algebras (and variations on the theme) are more suitable for specifying a completely defined system more than for refining a project or represent (parts of) a library of specifications *on the shelf.*

An opposite point of view is presented by Goguen in [43], where the focus is on the errors, that are considered as the primitive part of the specification and detected "at compile time", as terms (deduced equal to terms) where symbols from a selected part of the signature appear.

2.1.2 Equational Type Logic

Equational type logic (see e.g. [59]) is an extension of conditional logic based on equality, that widens "equational reasoning" toward "reasoning with equations and type assignments". The intuition is to put together symbols to represent both types and their elements in one carrier and use a "typing" predicate family to describe the belonging of elements to types. This approach allows to state equalities (or inclusions) between sorts (viewed as first class elements) and, representing a (total) many-sorted operation as a one-sorted function that on correctly typed inputs yields a correctly typed output, partial functions, too, that on correctly typed inputs can yield correctly typed outputs (but can as well yield an untyped result, representing an "undefined" or "non-terminating" computation, or an error).

Technically equational type logic (from now on ET-logic for short) is a particular subcase of usual first-order logic with equality where the only predicate is a binary *typing* relation and the sentences are positive Horn-Clauses. The technical simplicity, even poorness, of this approach on the logical side allows the definition of a very efficient deductive calculus, but on the semantic side makes the models of theories expressed in ET-logic difficult to figure and manage, with the elements of different types and the names of the sorts all together in one carrier, with a lot of junks due to the application of functions to inputs outside their domains.

Def. 2.1.3 The institution of *equational type logic* is the subinstitution $\mathcal{ET} = (\mathbf{Sign}_{\mathcal{ET}}, Sen_{\mathcal{ET}}, Mod_{\mathcal{ET}}, \models_{\mathcal{ET}})$ of $\mathcal{L}$, whose signature have just one binary predicate. $\Box$

In the sequel the binary predicate of any $\mathcal{ET}$-signature will be denoted by the symbol : with infix notation $\_ : \_$.

Consider again the example of the specification of natural numbers with the plus, times and minus operations.

Example 2.1.4

spec $Nat_{\mathcal{ET}}$ =
constants $N, zero$
opns
S *arity* 1
$+, *, -$ *arity* 2
axioms
α_1 $zero : N$
α_2 $x : N \supset S(x) : N$
α_3 $zero + x = x$
α_4 $S(x) + y = S(x + y)$
α_5 $x * zero = zero$
α_6 $S(x) * y = (x * y) + y$
α_7 $x - zero = x$
α_8 $S(x) - S(y) = x - y$

In the initial model of this specification the subset $Nat = \{n \mid n : N\}$ together with the restriction of the operations $+$, $*$ and $-$ to Nat is (a representation of) the natural numbers; however note that in the carrier there are elements to represent not only the "errors", like $zero - S(S(zero))$, but also "meaningless" expressions, like $zero - S(zero + N)$ or $N * N$.

The amount of junk in the carrier of $\mathcal{ET}$-models quickly increases if non-homogeneous data types are defined; for example defining the *stacks* of natural numbers enriching the above specification, terms intuitively non-well formed, as $S(pop(N) + empty)$, have to be interpreted.

2.1.3 Unified Algebras

The *unified algebras* by Mosses (see e.g. [68]) are based on the intuition that the values in the carrier represent not only *elements* of the data, but also *classifications* of elements into *sorts*, i.e. the elements of the carrier are values, set of values and names for these sets. To formalize the idea that the carriers of unified algebras are set of sets, the carriers are required to be distributive lattices with bottom and the lattice partial order represents sort inclusion, so that the bottom represents the "empty-sort".

The "empty-sort" is quite natural to represent the lack of result of partial operations as well as a "sort" that classifies several elements may be regarded as a *non-deterministic choice* between those elements, providing a tool to formalize non-deterministic results of a computation.

Having sort names as elements in the carrier also allows to apply functions to sorts, so that on one hand many useful classifications of elements can be expressed directly, without naming them by constants (for example the sort of positive natural number is represented by the application of the *successor* operation to the sort symbol *Nat* classifying the natural numbers) and on the other sort constructors of dependent or generic data types are just usual operations.

Def. 2.1.5 [[68]] The institution of *unified algebras* is the subinstitution $\mathcal{UNI} = (\mathbf{Sign}_{\mathcal{UNI}}, Sen_{\mathcal{UNI}}, Mod_{\mathcal{UNI}}, \models_{\mathcal{UNI}})$ of $\mathcal{L}$, where:

- $\mathbf{Sign}_{\mathcal{UNI}}$ is the subcategory of $\mathbf{Sign}_{\mathcal{L}}$ whose objects includes the constant symbol *nothing*, two binary operations $\_|\_$ and $\_\&\_$, and the two predicate symbols $\_ \leq \_$ and $\_ : \_$; the arrows are the signature morphisms preserving these symbols.

- $Sen_{\mathcal{UNI}}: \mathbf{Sign}_{\mathcal{UNI}} \to \mathbf{Set}$ is the restriction of $Sen_{\mathcal{L}}$ to $\mathbf{Sign}_{\mathcal{UNI}}$;

- $Mod_{\mathcal{UNI}}: \mathbf{Sign}_{\mathcal{UNI}}{}^{op} \to \mathbf{Cat}$ is defined by:

  - for every signature Σ in $\mathbf{Sign}_{\mathcal{UNI}}$ the category $Mod_{\mathcal{UNI}}(\Sigma)$ is the full subcategory of $Mod_{\mathcal{L}}(\Sigma)$ whose objects A satisfy the following conditions

    * the carrier of A is a *distributive lattice* with $\_|^A\_$ as join, $\_\&^A\_$ as meet and $nothing^A$ as bottom;
    * there is a distinguished set of values $E_A \subseteq A$ (the *elements* of A);
    * f^A is monotone w.r.t. the partial order of the lattice for all $f \in Op$;
    * $x \leq^A y$ holds iff $x|^A y = y$ (i.e. $\leq^A$ is the partial order of the lattice);
    * $x :^A y$ holds iff $x \in E_A$ and $x \leq^A y$.

  - for every signature morphism $\rho: \Sigma \to \Sigma'$, the functor $Mod_{\mathcal{UNI}}(\sigma)$ is the restriction of $Mod_{\mathcal{L}}(\sigma)$ to $Mod_{\mathcal{UNI}}(\Sigma')$.

- $A \models_{\mathcal{UNI}_\Sigma}$ is the restriction of $\models_{\mathcal{L}}$ to $|Mod_{\mathcal{UNI}}(\Sigma)| \times Sen_{\mathcal{UNI}}(\Sigma)$. $\square$

Note that the unified algebras are the model class of the following Horn-Clause set on the minimal unified signature.

spec Sp_{Uni} =

$$
\begin{array}{ll}
\mathit{nothing} \leq x & \\
x \leq y \wedge y \leq x \supset x = y & \\
x \leq y \wedge y \leq z \supset x \leq z & \\
x \leq x & \\
x \leq z \wedge y \leq z \supset x|y \leq z & \\
x \leq x|y & \\
y \leq x|y & \\
z \leq x \wedge z \leq y \supset z \leq x\&y & \\
x\&(y|z) = (x\&y)|(x\&z) & \\
x|(y\&z) = (x|y)\&(x|z) & \\
x : x \wedge x \leq y \supset x : y & \\
x : y \supset x : x & \\
x : y \supset x \leq y & \\
x_1 \leq x'_1 \wedge \ldots \wedge x_n \leq x'_n \supset f(x_1, \ldots, x_n) \leq f(x'_1, \ldots, x'_n) & \text{for all } f \in Op_n
\end{array}
$$

Therefore the results about initiality and deduction of homogeneous Horn-Clauses logic also hold for unified theories.

From a practical point of view, specifications in the unified algebra frameworks are equivalent to the ones in the $\mathcal{ET}$-logic, in the sense that the logical side is quite simple (although the axioms of Sp_{Uni} are implicit in any theory and must be carefully taken in account to correctly specify data types), but the models of specifications have carriers full of junk.

2.2 The Order-Sorted Approach

The order-sorted approach adds, as its name states, a partial order on the sorts, that reflects on the semantic side as an inclusion between the correspondent carriers and in particular modifies the definition of terms, having that if t is a term of sort s and s is a subsort of s', then obviously t is a term of sort s' too.

The overloading of function symbols, common to most algebraic frameworks and (meta)programming practice, gets a new flavor interacting with the order-sorted feature. Consider indeed two paradigmatic examples:

- the integer numbers are included (as subtype) by the rational numbers; in this case functions like sum and product are defined twice: on the integers and on the rational numbers; but obviously the interpretation of these operations for the rational case should agree with the interpretation for the integers on integers values.

- booleans with *and*, denoted by $*$, and *or*, denoted by $+$, are interpreted by the natural numbers subset $\{0,1\}$ in the usual way; in this case there is no reason to require that the interpretation of $*$ or $+$ on the natural numbers restricted to the subsort of boolean values coincides with the boolean interpretation of $*$ or $+$.

Since most concrete cases follow the pattern of the first example, in the original papers on order-sorted Goguen and Meseguer (see e.g. [39, 40]) restricted signatures to the *monotonic* ones, where if both $f: s_1, \ldots, s_n \to s$ and $f: s'_1, \ldots, s'_n \to s'$ for some $s_i \leq s'_i$ for $i = 1 \ldots n$, then $s \leq s'$ and accordingly algebras to those where function symbols on subsorts are interpreted by the restriction of the interpretations on the sorts, i.e., using the above example, where ${f_{s'_1,\ldots,s'_n,s'}}^A(a_1, \ldots, a_n) = {f_{s_1,\ldots,s_n,s}}^A(a_1, \ldots, a_n)$ for all $a_i \in {s_i}^A$ and $i = 1 \ldots n$. More recently Goguen and Diaconescu in [47] relaxed this condition by allowing a subsignature to be declared non-monotonic.

A more model-theoretic approach to overloading in the order sorted paradigm (see e.g. [37, 73, 88]) consists of requiring that the interpretations of the same function symbol, disregarding its functionality, produces the same output if provided with the same input values, i.e. if $f: s_1, \ldots, s_n \to s$ and $f: s'_1, \ldots, s'_n \to s$, then ${f_{s_1,\ldots,s_n}}^A(a_1, \ldots, a_n) = {f_{s'_1,\ldots,s'_n}}^A(a_1, \ldots, a_n)$ for all $a_i \in ({s_i}^A \cap {s'_i}^A)$, or equivalently each function symbols is interpreted by a partial function on a universe, including the carriers of the algebra, s.t. if $f: s_1, \ldots, s_n \to s$, then ${s_1}^A \times \ldots \times {s_n}^A$ is in the domain of f^A and $f({s_1}^A \times \ldots \times {s_n}^A) \subseteq s^A$. This approach is contrary to the abstractness of usual algebraic frameworks, where carriers are considered equivalent if (set-theoretically) isomorphic.

In order to have that the order-sorted terms with the usual (unparsed) functional notation can be given a unique interpretation, the concept of *least sort* of a term is introduced and in order to the least sort exists for every term, the *regularity* condition on the signature is required. Finally to have that validity is invariant under algebra isomorphism (that seems quite a reasonable requirement for any algebraic framework), the signature has to be *coherent*.

Def. 2.2.1 An *order sorted signature* is a triple $(S, \leq, F)$ s.t. (S, F) is a many-sorted signature and $(S, \leq)$ is a poset. In the sequel the symbol $\leq$ is also used to denote the extension of the partial order $\leq$ on strings of sorts ($s_1 \ldots s_n \leq s'_1 \ldots s'_{n'}$

iff $n = n'$ and $s_i \leq s'_i$ for $i = 1 \ldots n$) and on pairs (w, s), where $w \in S^*$ and $s \in S$ ($(w, s) \leq (w', s')$ iff both $w \leq w'$ and $s \leq s'$).

An order sorted signature $\Sigma = (S, \leq, F)$ is *monotone* iff $f \in F_{w,s} \cap F_{w',s'}$ and $w \leq w'$ implies $s \leq s'$. Σ is *regular* iff it is monotone and given $f \in F_{w_1,s_1}$ and $w_0 \leq w_1$, there is a least rank (w, s) s.t. $w_0 \leq w$ and $f \in F_{w,s}$; Σ is *coherent* iff it is regular and each connected component of $(S, \leq)$ (equivalence class of the symmetric and transitive closure of $\leq$) is *filtered* i.e. for any s, s' in the connected component of $\bar{s}$ there exists s'' in the same component s.t. $s, s' \leq s''$.

An *order sorted algebra* A on an order sorted signature $(S, \leq, F)$ is a many-sorted algebra on (S, F) s.t. $s^A \subseteq s'^A$ for all $s \leq s'$ in $(S, \leq)$. An *order sorted homomorphism* $h\colon A \to B$ between order sorted algebras on $(S, \leq, F)$ is a many-sorted homomorphism s.t. h_s is the restriction of $h_{s'}$ to s^A for all $s \leq s'$ in $(S, \leq)$.

An order sorted algebra A on $(S, \leq, F)$ is *monotone* on a function symbol f iff $f \in F_{w,s} \cap F_{w',s'}$ and $(w, s) \leq (w', s')$ imply that ${f_{w,s}}^A$ is the restriction of ${f_{w',s'}}^A$ to ${s_1}^A \times \ldots \times {s_n}^A$ for $w = s_1, \ldots, s_n$. $\square$

Then to represent a partial function, a sort has to be added to the signature, that denotes the *domain* of this function and is a subsort of the function source. A minor problem of this point of view is that to deal with non-unary functions, the set of sorts have to be closed under product formation (or at least under some products) and accordingly projection and tupling operations have to belong to the function symbol set, so that the syntax of types results heavier than necessary and the models carry redundant structure, unless some *ad hoc* solution is adopted, as in the following specifications of rational numbers.

Example 2.2.2 Consider the usual specification of integers and enrich it to define the rational numbers.

spec Int_{OSA} =
  sorts Z
  opns
    $0\colon \to Z$
    $s, p\colon Z \to Z$
    $\_ + \_, \_ - \_, \_ * \_\colon Z \times Z \to Z$
  axioms **Var** x, y:Z
    α_1 $p(s(x)) = x$
    α_2 $s(p(x)) = x$
    α_3 $0 + x = x$
    α_4 $s(x) + y = s(x + y)$
    α_5 $p(x) + y = p(x + y)$

α_6 $x - 0 = x$
α_7 $x - s(y) = p(x - y)$
α_8 $x - p(y) = s(x - y)$
α_9 $0 * x = 0$
α_{10} $s(x) * y = (x * y) + y$
α_{11} $p(x) * y = (x * y) - y$

In order to be able to "statically" distinguish if an integer is not zero, and hence can be used as denominator to build rational numbers, the sort of (strictly) positive integers has to be introduced, contrary to the modularity principles.

spec Rat_{OSA} =
enrich Int_{OSA} **by**
sorts $N_+ \leq N \leq Z \leq Q$
opns
$0 \colon \to N$
$s \colon N \to N_+$
$\_/\_ \colon Z \times N_+ \to Q$
$\_ * \_ \colon N_+ \times N_+ \to N_+$
$\_ + \_, \_ - \_, \_ * \_ \colon Q \times Q \to Q$
axioms **Var** x, y:Z; $z, t : N_+$
β_1 $(x * t) = (z * y) \supset (x/z) = (y/t)$
β_2 $(x/z) + (y/t) = ((x * t) + (z * y))/(z * t)$
β_3 $(x/z) - (y/t) = ((x * t) - (z * y))/(z * t)$
β_4 $(x/z) * (y/t) = (x * y)/(z * t)$

Note that the operation $\_ * \_ \colon N_+ \times N_+ \to N_+$ is introduced just in order to the terms in axioms β_2, β_3 and β_4 are well-formed. This is a quite common situation, using order-sorted specifications, that an operation is defined also on sub-sorts to express that the results on restricted inputs are typed in a smaller sort.

Although the "static" check has its fascination, it is worth to note that terms intuitively well-formed are inadmissible; consider indeed in the above specification the string $0/(s(0) + s(0))$; this is not a well formed term, because $s(0) + s(0)$ has type Z instead of N_+, although $s(0) + s(0)$ is reducible, by α_3 and α_4 to $s(s(0))$ that has the correct type N_+. To solve this problem *retracts* are used, that "try" to apply a more restrictive typing to terms. Formally retracts are unary functions from a supsort into a subsort, in the above example the retract would be $r_{Z,N_+} \colon Z \to N_+$, and a term on the enriched signature is interpreted as a value iff it reduces to a term on the original signature, as in the example above the term $0/r_{Z,N_+}(s(0) + s(0))$ that reduces to $0/s(s(0))$, otherwise it can be seen as an error, as $0/r_{Z,N_+}(0)$ that cannot be further reduced. Thus the original elegance of the order-sorted approach is reduced to another variation on the theme of error algebras.

A classical example where the standard approach with product sorts is more comfortable is the specification of the minus operation on natural numbers.

Example 2.2.3 In the following specification besides the data type operations 0, successor, +, and minus, the projections and tupling are introduced to handle the product sort and a constructor of the domain of the minus; the axioms defining the mutual behaviour of projections and tupling are standard not only in this example, but also in the general case, so that they could be introduced automatically as well as the operations themselves.

spec *Minus* =
  sorts $N, D \leq N \otimes N$
  opns
    $0 \colon \to N$
    $s \colon N \to N$
    $\_ + \_ \colon N \times N \to N$
    $minus \colon D \to N$

    $< \_, \_ > \colon N \times N \to N \otimes N$
    $\pi_1 \colon N \otimes N \to N$
    $\pi_2 \colon N \otimes N \to N$

    $constr \colon N \times N \to D$
  axioms **Var** x, y:N
    β_1 $\pi_1(< x, y >) = x$
    β_2 $\pi_2(< x, y >) = y$
    β_3 $< \pi_1(z), \pi_2(z) >= z$

    α_1 $0 + x = x$
    α_2 $s(x) + y = s(x + y)$
    α_3 $constr(x, y) =< x + y, y >$
    α_4 $minus(constr(x, y)) = x$

If $n \geq m$, then $< n, m >= constr(n - m, m)$ and hence *constr* builds the set $\{(n, m) \mid m \leq n\}$, intended domain of the minus operation, but it seems quite counterintuitive to express for example $s(s(s(0))) - s(0)$ by $minus(constr(s(s(0)), s(0)))$.

It is also worth to note that partial functions whose domains are "dependent" types are not supported by the order-sorted approach. Consider indeed the specification of the arrow composition in a category; then $f; g$ is defined iff the target of f matches the source of g. Apparently there is no way of defining the domain of the composition in the order-sorted style. The solution in similar cases is to (possibly automatically) introduce supersorts of "possibly erroneous elements" and duplicate the operations on the new sorts too and then to forget/hide this "error" part. But, since there is no explicit type declaration mechanism, the use of error supersort not always works in combination with loose semantics. In particular in the case of categories there is no way to impose that the composition of correctly composable arrows is in the subsort of (correct) arrows.

Example 2.2.4 Note that the introduction of errors on arrows to specify the composition also requires to introduce errors on objects (on any other sort related to them by an operation) to propagate the errors. Note also that there are no constants, so that the initial model is empty; the following is intended to be the specification of categories in the sense that (small) categories satisfy this specification, but it has also non-standard models.

spec *Categories* =
    sorts $obj \leq obj_{Err}, arr \leq arr_{Err}$
    opns
        $id\colon obj \to arr$
        $\delta_0, \delta_1\colon arr \to obj$
        $\_;\_\colon arr_{Err} \times arr_{Err} \to arr_{Err}$

        $\delta_0, \delta_1\colon arr_{Err} \to obj_{Err}$
    axioms   **Var** $A{:}obj;\ f, g, h : arr_{Err}$
        $\alpha_1 \quad \delta_0(id(A)) = A$
        $\alpha_2 \quad \delta_1(id(A)) = A$
        $\alpha_3 \quad \delta_0(f; g) = \delta_0(f)$
        $\alpha_4 \quad \delta_1(f; g) = \delta_1(g)$
        $\alpha_5 \quad (f; g); h = f; (g; h)$
        $\alpha_6 \quad id(\delta_0(f)); f = f$
        $\alpha_7 \quad f; id(\delta_1(f)) = f$

2.3 The Partial Approach

The theory of partial algebras has been developed by different groups, so that a common notation is missing and sometimes there are subtle differences between apparently equivalent definitions. The notation here coincides more or less with that used by Goguen and Meseguer in [64] for the total case and Broy and Wirsing in [23]. For general and exhaustive expositions on the subject see [26, 80]; here the main concepts and results are summarized in order to support a deep discussion on a more general kind of partial specifications.

In the following the symbol $=$ will always denote strong equality, i.e. if p and q are expressions in the metalanguage, then $p = q$ holds iff either both p and q are undefined, or both are defined and equal.

Partial algebras differ from many-sorted total ones, because the operation symbols are interpreted by partial functions, i.e. by functions whose domains are (possibly proper) subsets of their sources.

Def. 2.3.1

- A *partial algebra* A on a many-sorted signature $\Sigma = (S, F)$ consists of a family $\{s^A\}_{s \in S}$ of sets, the *carriers*, and of a family $\{op^A\}_{op \in F_{w,s}, w \in S^*, s \in S}$ of partial functions, the *interpretations of operation symbols*, s.t.

  - if $w = \Lambda$ then either op^A is undefined or $op^A \in s^A$;
  - if $w = s_1 \ldots s_n$, where $n \geq 1$, then $op^A : {s_1}^A \times \ldots \times {s_n}^A \xrightarrow{p} s^A$.

  Often the partial algebra A is denoted by the couple $(\{s^A\}, \{op^A\})$, omitting the quantifications about s and op which are associated with the signature. A partial algebra over a signature Σ is called a Σ-algebra. The class of all Σ-algebras is denoted by $PA(\Sigma)$.

- Let A be a partial algebra on a signature $\Sigma = (S, F)$; then a Σ- algebra B is a *subalgebra* (regular subobject) of A iff $s^B \subseteq s^A$ for all $s \in S$ and $op^B(b_1, \ldots, b_n) = op^A(b_1, \ldots, b_n)$ for all $op \in F_{s_1 \ldots s_n s}$ and all $b_i \in {s_i}^B$ for $i = 1 \ldots n$. □

As in the many-sorted total case, in the sequel the constant $op \in F_{\Lambda,s}$ is regarded as a "zeroary" operation, with the convention that ${s_1}^A \times \ldots \times {s_0}^A$ is the singleton set, so that a partial function having it as domain is either undefined on an element of the codomain, accordingly with the definition of op^A.

The definition of homomorphism adopted here, well known in literature, see e.g. [26, 22] (where it is called total Σ- homomorphism), is used in the initial and loose approaches, because, preserving existential equalities, makes the initial object in a class, if any, to satisfy the no-junk and no-confusion conditions (rephrased with existential equalities) and accordingly structures the model class.

Def. 2.3.2 Let A and B be two Σ-algebras and p be a family of total functions $p = \{p_s: s^A \to s^B\}_{s \in S}$. Then p is a *homomorphism* from A into B iff for any $op \in F_{s_1 \dots s_n, s}$, with $n \geq 0$, and any $a_i \in {s_i}^A$ with $i = 1 \dots n$:
$op^A(a_1, \dots, a_n) \in s^A$ implies $p_s(op^A(a_1, \dots, a_n)) = op^B(p_{s_1}(a_1), \dots, p_{s_n}(a_n))$.

In the following any homomorphism p from A into B is denoted by $p: A \to B$.

For each signature $\Sigma = (S, F)$ the category $\mathbf{PAlg}(\Sigma)$ of partial Σ-algebras consists of:

- the set of objects of $\mathbf{PAlg}(\Sigma)$ is $PA(\Sigma)$;
- $\mathbf{PAlg}(\Sigma)(A, B) = \{p: A \to B \mid p \text{ homomorphism}\}$ for all $A, B \in PA(\Sigma)$;
- the *identity* morphism Id_A is $Id_A = \{Id_{s^A}\}_{s \in S}$, where Id_{s^A} is defined by $Id_{s^A}(a) = a$ for all $a \in s^A$ and all Σ-algebras A;
- let A, B and C be Σ-algebras, $p: A \to B$ and $q: B \to C$ be homomorphisms; then $q \circ p$ is the family $\{q_s \circ p_s\}_{s \in S}$.

Let C be a class of Σ-algebras and X be a family of variables s.t. there exists at least an $A \in C$ and a valuation for X in A. A couple (Fr, m), where Fr is a Σ-algebra and m is a valuation for X in Fr, is *free* over X in C iff

- $Fr \in C$;
- for all $A \in C$ and all valuations V for X in A, there exists a unique homomorphism p_V from Fr into A s.t. $p_V(m(x)) = V(x)$ for all $x \in X$.

Let C be a class of Σ-algebras; an algebra I is initial in C iff it is free over the empty set in C, i.e. iff $I \in C$ and for all $A \in C$ there exists a unique homomorphism from I into A. $\square$

Note that the definition of free object coincide with the usual definition in category theory, for free object generated by X w.r.t. the forget functor from $\mathbf{PAlg}(\Sigma)$ into $\mathbf{Set}_S$ (the category of S-sorted sets), see e.g. [54].

The natural interpretation of (usual total) terms in a partial algebra is defined as in the total case substituting each variable for its valuation and each operation symbol for its interpretation in the algebra. It is worth to note that the natural interpretation is *strict* in the sense that if the natural interpretation of a subterm is undefined, then the natural interpretation of the term is undefined, too, because the functions of algebras are strict, too.

Def. 2.3.3 Let A be a Σ-algebra, $X = \{X_s\}_{s \in S}$ be an S-sorted family of variables and $V = \{V_s: X_s \to s^A\}_{s \in S}$ be a family of total functions, called a *valuation* for X in A.

- Then the *natural interpretation* of terms w.r.t. A and V, denoted by $eval^{A,V}$, is the partial function inductively defined by the following clauses, where $t^{A,V}$ stands for $eval^{A,V}(t)$:
  - $x^{A,V} = V_s(x)$, for all $x \in X_s$ and $op^{A,V} = op^A$, for all $op \in F_{\Lambda,s}$;
  - $(op(t_1, \ldots, t_n))^{A,V} = op^A({t^{A,V}}_1, \ldots, {t^{A,V}}_n)$ for all $op \in F_{s_1 \ldots s_n, s}$, with $n \geq 1$, and all $t_i \in T_\Sigma(X)_{|s_i}$.

  When restricted to T_Σ, $eval^{A,V}$ is denoted by $eval^A$ and, correspondingly, $t^{A,V}$ becomes t^A.

- The *term-image* algebra $V(A)$ is the subalgebra of A defined by:

$$s^{V(A)} = \{a \mid \text{ there exists } t \in T_\Sigma(X)_{|s} s.t.\ a = t^{A,V}\} \text{ for all } s \in S.$$

- The *kernel* of the natural interpretation of terms w.r.t. A and V, denoted by $K^{A,V}(X)$ or simply by K^A if X is the empty set, is the family $\{K^{A,V}(X)_s\}_{s \in S}$ where

$$K^{A,V}(X)_s = \{(t, t') \mid t, t' \in T_\Sigma(X)_{|s}, t^{A,V}, t'^{A,V} \in s^A \text{ and } t^{A,V} = t'^{A,V}\}.$$

- If $eval^A$ is surjective, then A is called *term-generated.* $\square$

As in the total case homomorphisms preserve the evaluation of terms and this property is essential to have that the no-confusion property holds for the initial (free) model in a class, if any.

Prop. 2.3.4 Let A and B be two Σ-algebras, X be a family of variables, V_A and V_B be two valuations for X in A and B resp.

1. If $p: A \to B$ is a homomorphism s.t. $p(V_A(x)) = V_B(x)$ for all $x \in X$, then $p(t^{A,V_A}) = t^{B,V_B}$ for all $t \in T_\Sigma(X)_{|s}$ s.t. $t^{A,V_A} \in s^A$.
2. If $eval^{A,V_A}$ is surjective, then there exists at the most one homomorphism $q: A \to B$ s.t. $q(V_A(x)) = V_B(x)$ for all $x \in X$.

Proof. By structural induction over $T_\Sigma(X)$. $\square$

In order to show that the initial model in a class, if any, satisfies the no-junk condition, i.e. that any its element is denoted by some term, it is useful to introduce the notions of congruence and quotient.

Def. 2.3.5

- Given a signature $\Sigma = (F, S)$ and a Σ-algebra A, a *congruence* $\equiv$ over A is a family of binary relations $\{\equiv_s\}_{s \in S}$ satisfying the following conditions (where the obvious quantifications over sorts is omitted):
  - $\equiv_s \subseteq s^A \times s^A$, and $\equiv_s$ is symmetric and transitive; in the following the set $\{a \mid a \equiv_s a\}$ is denoted by $Dom(\equiv_s)$ and $a {\equiv_s}^D a'$ holds iff either $a \equiv_s a'$ or $a, a' \notin Dom(\equiv_s)$;
  - for all $op \in F_{s_1 \ldots s_n, s}$ and all $a_i, a'_i \in {s_i}^A$, with i = 1… n, if $a_i \equiv_{s_i} a'_i$ for $i = 1 \ldots n$, then $op^A(a_1, \ldots, a_n) {\equiv_s}^D op^A(a'_1, \ldots, a'_n)$.
  - for every $op \in F_{s_1 \ldots s_n, s}$ and every $a_i \in {s_i}^A$, with $i = 1 \ldots n$, if $op^A(a_1, \ldots, a_n) \in Dom(\equiv_s)$, then $a_i \in Dom(\equiv_{s_i})$, for $i = 1 \ldots n$.
- Let $\equiv$ be a congruence over a Σ-algebra A; let $[a]$ denote the equivalence class of a in $\equiv_s$ for all $s \in S$ and all $a \in s^A$. The *quotient algebra* of A w.r.t. $\equiv$, denoted by $A/\equiv$, is defined by:
  - $s^{A/\equiv} = \{[a] \mid a \in Dom(\equiv_s)\}$, for all $s \in S$;
  - $op^{A/\equiv}([a_1], \ldots, [a_n]) = [op^A(a_1, \ldots, a_n)]$ if $op^A(a_1, \ldots, a_n) \in Dom(\equiv_s)$, otherwise $op^{A/\equiv}([a_1], \ldots, [a_n])$ is undefined, for all $op \in F_{s_1 \ldots s_n, s}$ and all $a_i \in Dom(\equiv_{s_i})$ with $i = 1 \ldots n$. □

Prop. 2.3.6 Let A be a Σ-algebra, $\equiv$ be a congruence, X be a family of variables and V, V' be valuations respectively for X in A and for X in $A/\equiv$ s.t. $V'(x) = [V(x)]$. Then $[t^{A,V}] = t^{A/\equiv,V'}$ for every term $t \in T_\Sigma(X)_{|s}$.
Proof. By structural induction over $T_\Sigma(X)$. □

The Prop. 2.3.6 implies, in particular, that if $eval^{A,V}$ is surjective, then $eval^{A/\equiv,V'}$ is surjective, too; thus in particular, for $A = T_\Sigma(X)$ and V the embedding, every quotient of a term-algebra is term-generated on the same variable set.

Prop. 2.3.7 Let A be a Σ-algebra and V be a valuation for a family X of variables in A.

1. The kernel $K^{A,V}(X)$ is a congruence over $T_\Sigma(X)$.

2. The algebras $V(A)$ and $T_\Sigma(X)/K^{A,V}(X)$ are isomorphic.

Proof.

1. The proof easily follows from the definition of $K^{A,V}(X)$.

2. It is easy to check that $p: T_\Sigma(X)/K^{A,V}(X) \to V(A)$, defined by $p_s([t]) = t^{A,V}$, and $q: V(A) \to T_\Sigma(X)/K^{A,V}(X)$, defined by $q_s(a) = [t]$, where $t \in T_\Sigma(X)_{|s}$ and $t^{A,V} = a$, are homomorphisms and that both $p \circ q = Id_{V(A)}$ and $q \circ p = Id_{T_\Sigma(X)/K^{A,V}(X)}$ by definition of p and q. $\square$

Note that in particular if A is generated by X via V, i.e. $eval^{A,V}$ is surjective, then A is isomorphic to a quotient of $T_\Sigma(X)$.

Prop. 2.3.8 Let A be a Σ-algebra and $\equiv_1$, $\equiv_2$ be congruences over A. The function $M: A/\equiv_1 \to A/\equiv_2$, defined by $M([a]_1) = [a]_2$, is a homomorphism iff $\equiv_1 \subseteq \equiv_2$.
Proof. The proof easily follows from the definitions of congruence and homomorphism. $\square$

The above proposition suggests that if the initial object of a class is term-generated, and hence isomorphic to the quotient of the term algebra by the kernel of the natural interpretation, then its kernel is contained in the kernel of any algebra of the class, i.e. it is the intersection of all kernels. It is easy to check that intersection of congruences is a congruence.

Def. 2.3.9 Let C be a class of Σ-algebras and X be a family of variables s.t. there exists at least an $A \in C$ and a valuation for X in A.

- For every family $\equiv = \{\equiv^i\}_{i \in I}$ of congruences over a Σ-algebra A the *intersection of* $\equiv$, denoted by $\cap(\equiv)$, is the congruence $\{\cap_{i\in I} {\equiv^i}_s\}_{s \in S}$.

- $K^C(X)$ is the intersection of the family
$$\{K^{A,V}(X) \mid A \in C, V: X \to A\}.$$
If X is the empty set, $K^C(X)$ is simply denoted by K^C. Moreover $T_\Sigma(X)/K^C(X)$ is denoted by $Fr^C(X)$ and the valuation $m^C: X \to Fr^C(X)$, defined by $m^C(x) = [x]_{K^C(X)}$, by m^C.

- $Gen(C, X)$ is the subclass of C defined by:
$$\{A \mid A \in C, \text{ there exists } V: X \to A \; s.t. \; eval^{A,V}(T_\Sigma(X)) = A\}.$$
If X is the empty set, then $Gen(C, X)$ is shortly denoted by $Gen(C)$. $\square$

Under minimal assumptions about the closure w.r.t. subalgebras and isomorphisms, existence and characterization of the free model for a class C of algebras can be stated independently from the fact that C is the model class of a set of (some kind of) sentences, because the nature of the free object is obliged by the definition of homomorphism.

Prop. 2.3.10 Let X be a family of variables and C be a class of Σ-algebras closed w.r.t. sub-algebra and isomorphism s.t. there exists at least an $A \in C$ and a valuation for X in A. The following conditions are equivalent:

1. there exists a free object for X in C;
2. $Fr^C(X)$ belongs to C;
3. $(Fr^C(X), m^C)$ is free for X in C;
4. there exists a free object for X in $Gen(C, X)$.

Proof.

1→2 Let (Fr, m) be the free object for X in C. Since C is closed w.r.t. sub-algebra, $m(Fr) \in C$ and hence $T_\Sigma(X)/K^{Fr,m}(X) \in C$, too, due to Prop. 2.3.7, because C is closed w.r.t. isomorphism. Therefore, in order to show that $Fr^C(X) \in C$, it is sufficient proving that $K^C(X) = K^{Fr,m}(X)$. By definition $K^C(X) \subseteq K^{Fr,m}(X)$, thus just $K^{Fr,m}(X) \subseteq K^C(X)$ has to be shown. Assume that $(t, t') \in K^{Fr,m}(X)$, i.e. $t^{Fr,m} = t'^{Fr,m} \in Fr$ and consider any $V: X \to A$ with $A \in C$; by definition of free object, $p_V: Fr \to A$ exists s.t. $p_V \circ m = V$ and hence, by Prop. 2.3.4(1), $t^{A,V} = p_V(t^{Fr,m})$ and $t'^{A,V} = p_V(t'^{Fr,m})$, so that from $t^{Fr,m} = t'^{Fr,m} \in Fr$ also $t^{A,V} = t'^{A,V} \in A$ follows.

2→3 Let A belong to C, $V: X \to A$ be a valuation and define $p_V: Fr^C(X) \to A$ by $p_V([t]) = t^{A,V}$; p_V is a function because $K^C(X) \subseteq K^{A,V}(X)$ and obviously is a homomorphism. Finally p_V is unique because of Prop. 2.3.4.

3→4 Since $Fr^C(X) \in Gen(C, X)$ by definition and $(Fr^C(X), m^C)$ is free for X in C, then $(Fr^C(X), m^C)$ is also free for X in $Gen(C, X)$.

4→2 Let (Fr, m) be the free object for X in $Gen(C, X)$; then $Fr^{Gen(C,X)}(X) \in Gen(C, X) \subseteq C$, because of 1→2 applied to $Gen(C, X)$. To show that $Fr^C(X) = Fr^{Gen(C,X)}(X)$ it is sufficient to show that $K =$

K_{Gen}, where $K = \{K^{A,V}(X) \mid A \in C, V\colon X \to A\}$ and $K_{Gen} = \{K^{A,V}(X) \mid A \in Gen(C,X), V\colon X \to A\}$, so that $K^C(X) = K^{Gen(C,X)}(X)$. Since $Gen(C,X) \subseteq C$, $K_{Gen} \subseteq K$. On the contrary let A belong to C and V be a valuation for X in A; then $V(A) \in C$, because C is closed w.r.t. sub-algebras, and hence $V(A) \in Gen(C,X)$, so that $K^{A,V} = K^{V(A),V} \in K_{Gen}$; thus $K \subseteq K_{Gen}$. Therefore $K = K_{Gen}$ and hence $Fr^C(X) = Fr^{Gen(C,X)}(X)$, so that $Fr^C(X) \in C$, because $Fr^{Gen(C,X)}(X) \in C$.

3→1 Obvious. □

The above proposition can be specialized for the initial case.

Cor. 2.3.11 Let C be a class of Σ-algebras closed w.r.t. sub-algebra and isomorphism. The following conditions are equivalent:

1. there exists an initial object in C;
2. T_Σ/K^C belongs to C;
3. T_Σ/K^C is initial in C;
4. there exists an initial object in $Gen(C)$

Proof. From Prop. 2.3.10, for $X = \emptyset$. □

The partial logic is built on two notions of equality: the existential equality, denoted by $=_e$, that holds iff both sides represent the same element of the carrier, and the strong equality, denoted by $=$, that holds iff either both sides are undefined or the existential equality holds. In the sequel, since the hypothesis is not restrictive from an applicative point of view and simplifies the proofs of some technical lemmas below, variables and both sorts and functions are assumed to be denumerable (at the most). In particular a denumerable universe *Var* of variables is assumed to be fixed and the families of variables used in the following statements are contained in *Var*.

Def. 2.3.12 Let $\Sigma = (S, F)$ be a signature and X be a family of S-sorted variables.

- An *elementary formula* over Σ and X has the form either $D(t)$ or $t{=}t'$ for $t, t' \in T_\Sigma(X)_{|s}$, where D denotes the definedness predicate (one for each sort; but sorts are omitted). The set of all elementary formulas over Σ and X will be denoted by $\mathrm{EForm}(\Sigma, X)$.

- A *conditional formula* over Σ and X has the form $\Delta \supset \epsilon$, where Δ is a countable set of elementary formulas over Σ and X and ϵ is an elementary formula over Σ and X too. The set of all conditional formulas on Σ and X will be denoted by $SC(\Sigma, X)$.

  If Δ is the empty set, then $\Delta \supset \epsilon$ is an equivalent notation for the elementary formula ϵ.

- A *positive conditional formula* over Σ and X is a conditional formula $\Delta \supset \epsilon$ over Σ and X s.t. $D(t)$ or $D(t')$ belongs to Δ for every $t{=}t'$ belonging to Δ. The set of all positive conditional formulas on Σ and X will be denoted by $Cond(\Sigma, X)$.

- For every formula ϕ let $Var(\phi)$ denote the set of all variables which appear in ϕ. A formula ϕ is called *ground* iff $Var(\phi)$ is empty.

- If A is a partial algebra, ϕ is a formula and V is a valuation for $Var(\phi)$ in A, then ϕ *holds* for V in A (equivalently: *is satisfied for V by A*) and write $A{\models_{\mathcal{PAR}}}_V\phi$ accordingly to the following:

  - $A{\models_{\mathcal{PAR}}}_V D(t)$ iff $t^{A,V}$ is defined; $A{\models_{\mathcal{PAR}}}_V t{=}t'$ iff $t^{A,V}$ and $t'^{A,V}$ are either both defined and equal or both undefined;
  - let ϕ be $\Delta \supset \epsilon$; then $A{\models_{\mathcal{PAR}}}_V\phi$ iff $A{\models_{\mathcal{PAR}}}_V\epsilon$, or $A\,{\not\models_{\mathcal{PAR}}}_V\delta$ for some $\delta \in \Delta$;

  For any formula ϕ, ϕ *holds in* (equivalently: *is satisfied by, is valid in*) A, denoted by $A\models_{\mathcal{PAR}}\phi$, iff $A{\models_{\mathcal{PAR}}}_V\phi$ for all valuations V for $Var(\phi)$ in A. $\square$

Remark.

- From the definition of validity, in the particular case $t = x$, $A\models_{\mathcal{PAR}}D(x)$ for all variables x follows, because $V(x) = x^{A,V}$ is defined for every valuation V, valuations being total functions.

- Note that $D(t)$ can be equivalently expressed by $t{=_e}t$; hence elementary formulas are just equalities either strong or existential. Analogously, since $t{=_e}t'$ is logically equivalent to $D(t) \wedge t{=}t'$, positive conditional formulas are (first-order equivalent to) conditional formulas whose premises are just existential equalities.

- The above notion of validity is the usual one in the many-sorted case; however some comments can be helpful. If $Var(\phi)_s \neq \emptyset$ and $s^A = \emptyset$, then

$A \models_{\mathcal{PAR}} \phi$ holds; hence for any class C of algebras, $C \models_{\mathcal{PAR}} \phi$ iff $A \models_{\mathcal{PAR}} \phi$ for all $A \in C$ s.t. $Var(\phi)_s \neq \emptyset$ implies $s^A \neq \emptyset$. Thus if C contains an algebra with all carriers non-empty (as it will always happen in the sequel), then the notion of validity for the class coincides with the classical one; for example it could not be that both $C \models_{\mathcal{PAR}} \phi$ and $C \models_{\mathcal{PAR}} \neg\phi$ (but note that negation is not in the language). Finally it is also useful to emphasize that here it is possible to stay within a two-valued logic, since any conditional formula is always either true or false for a (total) valuation of its variables. □

In the following a generic elementary formula will be denoted by ϵ or η or γ or δ, while a generic conditional formula will be denoted by ϕ or θ or ψ; moreover for all conditional formulas $\phi = (\Delta \supset \epsilon)$ the set Δ is denoted by $prem(\phi)$ and ϵ by $cons(\phi)$; finally $\epsilon_1 \wedge \ldots \wedge \epsilon_n \supset \epsilon$ is the same as $\{\epsilon_1 \ldots \epsilon_n\} \supset \epsilon$ for all elementary formulas $\epsilon_1 \ldots \epsilon_n, \epsilon$.

Def. 2.3.13 The institution of *many-sorted partial algebras* with conditional axioms is the quadruple $\mathcal{PAR} = (\mathbf{Sign}_{\mathcal{PAR}}, Sen_{\mathcal{PAR}}, Mod_{\mathcal{PAR}}, \models_{\mathcal{PAR}})$, where:

- $\mathbf{Sign}_{\mathcal{PAR}}$ is the category of many-sorted signatures $\mathbf{Sign}_{\mathcal{MS}}$.

- let X be a denumerable set of variables; for every signature Σ the set $Sen_{\mathcal{PAR}}(\Sigma)$ is $\{V.\xi \mid V: X \to S, \xi \in SC(\Sigma, Y) \text{ for } Y_s = V^{-1}(s)\}$. For every signature morphism $\rho: \Sigma \to \Sigma'$, where $\rho = (\sigma, \phi)$, the translation $Sen_{\mathcal{PAR}}(\sigma)$ of a sentence $V.\xi$ is $V \circ \sigma.ren(\xi)$, where $ren(\xi)$ denotes the renaming of function symbols in ξ by ϕ, leaving unaffected the variables.

- $Mod_{\mathcal{PAR}}: \mathbf{Sign}_{\mathcal{PAR}}{}^{op} \to \mathbf{Cat}$ is defined by:

  - for every signature $\Sigma = (\sigma, \phi)$ the category $Mod_{\mathcal{PAR}}(\Sigma) = \mathbf{PAlg}(\Sigma)$ has partial algebras as objects and partial algebra homomorphisms as arrows;

  - for every signature morphism $\rho: \Sigma \to \Sigma'$, where $\rho = (\sigma, \phi)$, the translation $Mod_{\mathcal{PAR}}(\sigma)(A')$ of a partial algebra A' is the reduct $(\{\sigma(s)^A\}_{s \in S}, \{\phi(f)^A\}_{f \in F}$ and the translation of a morphism h is $\{h_{\sigma(s)}\}_{s \in S}$.

- For any $\Sigma \in |\mathbf{Sign}_{\mathcal{PAR}}|$, any $A \in |Mod_{\mathcal{PAR}}(\Sigma)|$ and any $(V.\xi) \in Sen_{\mathcal{PAR}}(\Sigma)$, $A \models_{\mathcal{PAR}} V.\xi$ iff $A \models_{\mathcal{PAR}} \xi$ accordingly with Def. 2.3.12 for $\xi \in SC(\Sigma, Y)$ and $Y_s = V^{-1}(s)$.

The institution of many-sorted partial algebras with positive conditional axioms is the quadruple $\mathcal{PPAR} = (\mathbf{Sign}_{\mathcal{PAR}}, Sen_{\mathcal{PPAR}}, Mod_{\mathcal{PAR}}, \models_{\mathcal{PAR}})$ with positive conditional axioms as sentences and the same signatures, models and validity relation as $\mathcal{PAR}$, i.e. $Sen_{\mathcal{PPAR}}(\Sigma) = \{V.\xi \mid \xi \in Cond(\Sigma, \{V^{-1}(s)\})\}$. □

Positive conditional axioms always have an initial object and their model class form a quasi-variety (see e.g. [92]); thus technically they are not so far away from the total many-sorted approach. Moreover positive conditional axioms are sufficiently powerful to define most data types used in computer science. Consider again the examples used in the previous sections to illustrate the characteristics of the different approaches and see their simple partial specifications.

Example 2.3.14 The following specification of natural numbers with the minus operation is given in a modular style, enriching the usual (total) specification of natural numbers *à la Peano* by the binary operations $+$, $*$ and $-$

spec Nat =
  sorts N
  opns
    $0\colon \to N$
    $s\colon N \to N$
    $+, *, -\colon N \times N \to N$
  axioms
    α_1 $D(0)$
    α_2 $D(s(x))$

    α_3 $0 + x = x$
    α_4 $s(x) + y = s(x + y)$
    α_5 $x * 0 = 0$
    α_6 $s(x) * y = (x * y) + y$
    α_7 $x - 0 = x$
    α_8 $s(x) - s(y) = x - y$

The initial model of this specification is (a representation of) the natural numbers. Note that the axiom α_2 states that the successor operation is a total function, while plus and times are not required to be total, although they are interpreted as total function in the initial object, because the elements of the initial object are (denoted by) terms of the form $s^n(0)$ and hence using axioms α_3 and α_4 (resp. α_5 and α_6) the application $s^n(0) + s^m(0)$ reduces to the *defined* term $s^k(0)$, for $k = n + m$, and analogously $s^n(0) * s^m(0)$ reduces to $s^k(0)$, for $k = n * m$. Axioms α_7 and α_8 make the application of the minus operation to terms $s^n(0)$ and $s^m(0)$, for $n \geq m$, reduce to $s^k(0)$, for $k = n - m$, accordingly with the intuition; but nothing is required on $s^n(0) - s^m(0)$ in the case $n < m$ (or on non standard elements), so that in the initial model, because of the no-junk condition, the result is undefined; however in the model class there are algebras where the result is defined and can be used as an "error message", as in the total style. Thus this can be seen as a minimal specification of the minus on natural numbers that can be further refined fixing which kind of error messages are wanted.

The following construction of rational numbers can be easily generalized to the construction of quotient field on any integral domain, provided that a tool to distinguish the additive identity is at hand; note that this tool is also needed to impose that the ring is an integral domain. In the more general construction the *positive* predicate below would be changed in a *non-zero* predicate.

Example 2.3.15 Consider the usual (total) specification of integers and enrich it to define the rational numbers.

spec Int =
 sorts Z
 opns
  $0\colon \to Z$
  $s, p\colon Z \to Z$
  $\_ + \_, \_ - \_, \_ * \_\colon Z \times Z \to Z$
 axioms
  δ_1 $D(0)$
  δ_2 $D(s(x))$
  δ_3 $D(p(x))$

  α_1 $p(s(x)) = x$
  α_2 $s(p(x)) = x$
  α_3 $0 + x = x$
  α_4 $s(x) + y = s(x + y)$
  α_5 $p(x) + y = p(x + y)$
  α_6 $x - 0 = x$
  α_7 $x - s(y) = p(x - y)$
  α_8 $x - p(y) = s(x - y)$
  α_9 $0 * x = 0$
  α_{10} $s(x) * y = (x * y) + y$
  α_{11} $p(x) * y = (x * y) - y$

In order to distinguish non zero numbers, that can be used as denumerator, the predicate *positive* should be axiomatized; since predicates are not part of this paradigm, this predicate is represented as a boolean function, whose truth only is stated.

spec *Rat* =
  enrich *Int* **by**
  sorts B, Q
  opns
    $T\colon \to B$
    $positive\colon Z \to B$
    $\_/\_\colon Z \times Z \to Q$
    $\_+\_, \_-\_, \_*\_\colon Q \times Q \to Q$
  axioms
    $\delta_4 \quad D(T)$
    $\delta_5 \quad positive(x) = T \supset D(y/x)$

    $\gamma_1 \quad positive(s(0)) = T$
    $\gamma_1 \quad positive(x) = T \supset positive(s(x)) = T$

    $\beta_1 \quad (x * t) = (z * y) \supset (x/z) = (y/t)$
    $\beta_2 \quad (x/z) + (y/t) = ((x * t) + (z * y))/(z * t)$
    $\beta_3 \quad (x/z) - (y/t) = ((x * t) - (z * y))/(z * t)$
    $\beta_4 \quad (x/z) * (y/t) = (x * y)/(z * t)$

Note that if z is a negative number, then x/z is deduced equal to $(0 - x)/(0 - z)$, by β_1 and the axiomatization of the operations on integers, and hence is defined too, because $(0 - z)$ is positive, and hence $positive(0 - z) = T$.

As a final comparative example, consider again the (loose) specification of categories.

Example 2.3.16 As in the order-sorted approach there are no constants, so that the initial model is empty; but in this case the models of the specification are all and only the (small) categories.

spec *Categories* =
  sorts obj, arr
  opns
    $id\colon obj \to arr$
    $\delta_0, \delta_1\colon arr \to obj$

$\_;\_ : arr \times arr \rightarrow arr$

axioms

$\delta_1 \quad D(\delta_0(f))$
$\delta_2 \quad D(\delta_1(f))$
$\delta_3 \quad D(id(A))$
$\delta_4 \quad D(\delta_1(f)) \wedge \delta_1(f) = \delta_0(g) \supset D(f;g)$
$\delta_5 \quad D(f;g) \supset \delta_1(f) = \delta_0(g)$

$\alpha_1 \quad \delta_0(id(A)) = A$
$\alpha_2 \quad \delta_1(id(A)) = A$
$\alpha_3 \quad D(f;g) \supset \delta_0(f;g) = \delta_0(f)$
$\alpha_4 \quad D(f;g) \supset \delta_1(f;g) = \delta_1(g)$
$\alpha_5 \quad (f;g);h = f;(g;h)$
$\alpha_6 \quad id(\delta_0(f)); f = f$
$\alpha_7 \quad f; id(\delta_1(f)) = f$

The δ axioms guarantee that source and target are total functions, that any object has its identity and that the composition is defined iff its arguments are composable.

2.4 Strong Partial Logic

Although the positive conditional axioms are sufficiently powerful to define most part of the more common data types, they fail to specify sets of partial functions. Consider, indeed, the following specification of the (finite) maps, that are widely used in computer science, for example to represent environments and memories.

Example 2.4.1 Let X and Y be the sorts of the source and target respectively of the maps that are being specified, given respectively by the specifications sp_X and sp_Y accordingly with the modular approach. The operations on maps supported by this module are the *update* of a map m by an association (x, y), that corresponds to the insertion of a new couple if x is not in the domain of m and to the real update if an old couple (x, z) were already in m, and the *query* of the Y value correspondent to an X "address" in a map m. On X a decidable equality must be (pre)defined in order to the query have sense, so the existence of $eq: X \rightarrow B$ is assumed, where B is the boolean sort with constants $T: \rightarrow B$ and $F: \rightarrow B$, s.t. for any terms x, y of type X the equality $eq(x, y)$ reduces either to T or to F.

spec *Maps* =
enrich sp_X, sp_Y **by**
sorts *map*

opns

$initial\_status\colon \to map$
$upd\colon X \times Y \times map \to map$
$query\colon map \times X \to Y$

axioms

$\delta_1 \quad D(initial\_status)$
$\delta_2 \quad D(update(x, y, m))$

$\alpha_1 \quad eq(x, z) = T \supset query(update(x, y, m), z) = y$
$\alpha_2 \quad eq(x, z) = F \supset query(update(x, y, m), z) = query(m, z)$

Since the axioms do not impose equalities on the sort *map*, there is no way to deduce the intuitive identity of maps that give the same answer on each X element. Informally an axiom of the form

$$\star \quad (query(m, x) = query(m', x) \forall x \in X) \supset m = m'$$

would be needed, but the $\star$ axiom is not positive conditional, because of two reasons:

- the quantification on x only involves the premises;
- the equality in the premises is strong.

The first point can be disposed of, at least for term generated models, by means of an infinitary conjunction in the premises:

$$\{query(m, x) = query(m', x) \mid x \in T_\Sigma\} \supset m = m'$$

(see also [66, 67]), but the equality remains strong.

The following sections are devoted to the exposition of the recent results in [1, 7, 29] regarding the properties of non-positive conditional specifications.

It is worth noting that (both positive and non-positive) partial conditional specifications are aimed to deal with partiality due to still progressing refinement of specifications (loose approach) or to non-terminating computations. The treatment of errors and exception handling for the partial framework is (at the author's knowledge) still unexplored.

2.4.1 Conditional Specifications

The above and similar examples, especially from the higher-order paradigm, that is now becoming a rather popular and useful tool in algebraic specifications (see [66, 67]) lead to consider non-positive conditional specifications.

Notation.

- A *conditional specification* is a theory in $\mathcal{PAR}$. A generic conditional specification will be denoted by sp; the formulas belonging to Ax are called the *axioms* of sp and usually denoted by α.
- A *positive conditional specification* is a theory in $\mathcal{PPAR}$, i.e. a conditional specification s.t. all its axioms are positive conditional formulas; a generic positive conditional specification will be usually denoted by PSp.
- For any conditional specification $sp = (\Sigma, Ax)$, $PMod(sp)$ denotes the object class of $\mathbf{Mod}_{\mathcal{PAR}}(sp)$, i.e.

  $$PMod(sp) = \{A \mid A \in PA(\Sigma), A \models_{\mathcal{PAR}} \alpha \text{ for all } \alpha \in Ax\};$$

  an algebra $A \in PMod(sp)$ is called a *model* of sp.
- For every conditional specification $sp = (\Sigma, Ax)$ and every family X of variables, $K(Sp, X)$ denotes the congruence $K^{PMod(sp)}(X)$, i.e. the intersection of all kernels of natural interpretations of $T_\Sigma(X)$ in a model of sp; as an abbreviation let $Fr(sp, X)$ denote $Fr^{PMod(sp)}(X)$ and $m(sp, X)$ denote the valuation $m^{PMod(sp)}(X)$ (often Fr and m will be used when Sp and X are clear from the context).
- For every conditional specification $sp = (\Sigma, Ax)$, $PGen(sp, X)$ denotes the class $Gen(PMod(sp), X)$; moreover if X is empty, $PGen(sp, X)$ is simply denoted by $PGen(sp)$. □

Note that $PMod(sp)$ is not empty for all conditional specifications sp, since the trivial (total) algebra Z, with singleton sets as carriers and the obvious (total) interpretations of function symbols, is always a model. Moreover the trivial algebra Z has all carriers non-empty, so that there exists a valuation for all families X in Z and hence $K(sp, X)$ is always well defined.

Prop. 2.4.2 For all conditional specifications sp the class $PMod(sp)$ is closed under sub-algebras and isomorphisms.

Proof. The closure under isomorphism easily follows from the definition of validity; thus just consider the closure under sub-objects. Let A belong to $PMod(sp)$ and B be a sub-algebra of A. Let α be an axiom and V be a valuation for $Var(\alpha)$ into B; then V is also a valuation for $Var(\alpha)$ into A. Moreover it is easy to check that $t^{A,V} = t^{B,V}$ for all $t \in T_\Sigma(Var(\alpha))$ and hence that $A \models_{\mathcal{PAR}_V} \epsilon$ iff $B \models_{\mathcal{PAR}_V} \epsilon$

for all $\epsilon \in \mathrm{EForm}(\Sigma, Var(\alpha))$. Therefore $A \models_{\mathcal{PAR}_V} \alpha$, because A is a model of sp, and hence $B \models_{\mathcal{PAR}_V} \alpha$. $\square$

Thus Prop. 2.3.10 can be instantiated on $PMod(sp)$.

Prop. 2.4.3 Let X be a family of variables and $sp = (\Sigma, Ax)$ be a conditional specification. The following conditions are equivalent:

1. there exists a free object for X in $PMod(sp)$;
2. $Fr(sp, X) \in PMod(sp)$;
3. $(Fr(sp, X), m(sp, X))$ is the free object for X in $PMod(sp)$;
4. there exists a free object for X in $PGen(Sp, X)$.

Proof. From Prop. 2.3.10, as $PMod(sp)$ is closed under sub-algebras and isomorphisms because of Prop. 2.4.2. $\square$

Contrary to the case of positive conditional specifications, in general the class of models of a conditional specification is not necessary closed under binary product, as the following example shows.

spec sp_1 =
  sorts s
  opns
    $a, b\colon \to s$
  axioms
    $a{=}b \supset D(a)$

Let A and B be the models of sp_1 defined by:

Algebra A =
  $s^A = \{1\}$
  a^A is undefined
  $b^A = 1$

Algebra B =
  $s^B = s^A$
  $a^B = 1$
  b^B is undefined

Then the algebra $A \times B$ consists of:

Algebra $A \times B =$

$s^{A \times B} = \{(1,1)\}$

$a^{A \times B} = (a^A, a^B)$ is undefined, because a^A is undefined

$b^{A \times B} = (b^A, b^B)$ is undefined, because b^B is undefined

Therefore $A \times B$ is not a model of sp_1, because $A \times B \models_{\mathcal{PAR}} a{=}b$, both a and b being undefined, but $A \times B \not\models_{\mathcal{PAR}} D(a)$. □

While in the case of positive conditional specifications the closures under isomorphism, subalgebra and products are sufficient to guarantee the existence of (free) initial objects, in general the model class of a conditional specification is not required to have initial object. Indeed consider again the above specification sp_1; since an initial model, if any, is minimally defined, if an algebra I is initial in $PMod(sp_1)$, then both a and b are undefined, because they are undefined respectively in A and B, and hence I is not a model of Sp_1. It is easy to see that more sophisticated specifications exist, that admit initial model, but not free model for non-empty X; consider indeed the following specification sp_2.

spec $sp_2 =$

sorts s

opns

$zero\colon \to s$

$f, Succ\colon s \to s$

axioms

$\alpha_1 \quad Succ(x){=}f(x) \supset D(Succ(x))$

$\alpha_2 \quad D(Succ(zero))$

$\alpha_3 \quad D(Succ(x)) \supset D(Succ(Succ(x)))$

The initial model I of Sp_2 consists of:

Algebra $I =$

$s^I = \mathbb{N}$

$zero^I = 0$

$Succ^I(a) = a + 1$

f^I is the totally undefined function

Let A and B be the models of Sp_2 and V_A, V_B the valuations for $X = \{x\}$ in A, B resp. defined by:

Algebra $A =$

$s^A = \mathbb{N} \cup \{\infty\}$

$zero^A = 0$

$Succ^A(a) = a + 1$ if$a \in \mathbb{N}$
$Succ^A(\infty)$ is undefined
$f^A(a) = \infty$
$V_A(x) = \infty$

Algebra $B =$
$s^B = s^A$
$zero^B = 0$
$Succ^B(b) = b + 1$ if$b \in \mathbb{N}$
$Succ^B(\infty) = \infty$
f^B is the totally undefined function
$V_B = V_A$

Because of Prop. 2.4.3, in order to show that sp_2 has not a free model for X it is sufficient to show that $Fr(sp_2, X) \notin PMod(sp_2)$. Since both $Succ(x)^{A,V_A}$ and $f(x)^{B,V_B}$ are undefined, $Succ(x), f(x) \notin Dom(K(sp_2, X))$. Therefore $Fr(sp_2, X) \models_{\mathcal{PAR}_{m(sp_2,X)}} Succ(x) = f(x)$ and $Fr(sp_2, X) \not\models_{\mathcal{PAR}_{m(sp_2,X)}} D(Succ(x))$ so that $Fr(sp_2, X)$ does not satisfy α_1 and hence is not a model of sp_2.

Since the existence of a free model of a specification sp for a family X of variables is equivalent to $Fr(sp, X) \in PMod(sp)$, by Prop. 2.4.3, conditions that guarantee that $Fr(sp, X)$ satisfies the axioms of sp are interesting. Since $Fr(sp, X)$ is a quotient of a term algebra, it satisfies a formula ϕ for a valuation V iff it satisfies an instantiation $\rho(\phi)$ for the valuation $m(sp, X)$, where ρ substitutes each variable x for a a representative of the congruence class $V(x)$. Therefore a specification sp has a free model for a family X of variables iff $Fr(sp, X)$ satisfies the instantiations of the axioms (on defined terms) for $m(sp, X)$.

Def. 2.4.4 Let $sp = (\Sigma, Ax)$ be a conditional specification, X be a family of variables, Fr denote $Fr(sp, X)$ and m denote $m(sp, X)$. The set SNF(sp,X), where *SNF* stands for *Semantic Naughty Formulas*, consists of all conditional formulas $\Delta \supset \epsilon$ over Σ and X s.t.

snf_1 $\Delta \supset \epsilon$ is $\alpha[t_y/y \mid y \in Var(\alpha)]$ for some $\alpha \in Ax$ and some $t_y \in T_\Sigma(X)$ s.t. $Fr \models_{\mathcal{PAR}_m} D(t_y)$ for all $y \in Var(\alpha)$;

snf_2 $Fr \models_{\mathcal{PAR}_m} \delta$ for all $\delta \in \Delta$;

snf_3 $Fr \not\models_{\mathcal{PAR}_m} \epsilon$. □

Some short notations are introduced in order to make the presentation simpler.

Notation. Let X and Y be two families of variables, K be a congruence over $T_\Sigma(Y)$ and V be a valuation for X in $T_\Sigma(Y)/K$. For every $x \in X$ a term $t_{V,x} \in dom(K)$ s.t. $V(x) = [t_{V,x}]$ is denoted by $t_{V,x}$, by $V(t)$ the term $t[t_{V,x}/x \mid x \in X]$ for every term t, by $V(\theta)$ the formula $\theta[t_{V,x}/x \mid x \in X]$ for every formula θ and by $V(\Gamma)$ the set $\{V(\gamma) \mid \gamma \in \Gamma\}$ for every set Γ of formulas.

Note that if $A = T_\Sigma(Y)/K$, then $t^{A,V}$ is the equivalence class of $V(t)$ in K, by Prop. 2.3.6, and that $A \models_{\mathcal{PAR}_V} \gamma$ iff $A \models_{\mathcal{PAR}_m} V(\gamma)$, where $m(x) = [x]_K$.

Theorem 2.4.5 For every specification sp and every family X of variables, the following conditions are equivalent:

1. there exists a free object for X in $PMod(sp)$;
2. $Fr(sp, X) \in PMod(sp)$;
3. $(Fr(sp, X), m(sp, X))$ is free for X in $PMod(sp)$;
4. there exists a free object for X in $PGen(Sp, X)$;
5. $\mathrm{SNF}(sp, \mathrm{X}) = \emptyset$.

Proof. $1 \Leftrightarrow 2 \Leftrightarrow 3 \Leftrightarrow 4$ Follows from Prop. 2.4.3. Thus only $2 \Leftrightarrow 5$ have to be shown. Let sp be the specification (Σ, Ax), Fr denote $Fr(sp, X)$ and m denote $m(sp, X)$.

$\Rightarrow$ Assume that ϕ satisfies snf_1 and snf_2 and show that ϕ does not satisfy snf_3. Because of snf_1, ϕ is $\alpha[t_y/y \mid y \in Var(\alpha)]$ for some $\alpha \in Ax$ and $t_y \in T_\Sigma(X)$ s.t. $Fr \models_{\mathcal{PAR}_m} D(t_y)$. Let the valuation V for $Var(\alpha)$ in Fr be defined by $V(y) = [t_y]$; note that V is well defined, because $Fr \models_{\mathcal{PAR}_m} D(t_y)$ by snf_1 and hence $[t_y] \in Fr$. Since Fr is a model of sp, $Fr \models_{\mathcal{PAR}_V} \alpha$, i.e. $Fr \models_{\mathcal{PAR}_m} V(\alpha) = \phi$. Therefore, since $Fr \models_{\mathcal{PAR}_m} \delta$ for all $\delta \in prem(\phi)$ because of snf_2, $Fr \models_{\mathcal{PAR}_m} cons(\phi)$, i.e. ϕ does not satisfy snf_3.

$\Leftarrow$ Let α be an axiom of sp and V be a valuation for $Var(\alpha)$ in Fr. Then for all $y \in Var(\alpha)$ $t_{V,y} \in Dom(K(sp, X))$, i.e. $Fr \models_{\mathcal{PAR}_m} D(t_{V,y})$ and hence $V(\alpha)$ satisfies snf_1. Thus, since $\mathrm{SNF}(sp, \mathrm{X})$ is empty, $V(\alpha)$ does not satisfy snf_2 or snf_3, i.e. either $Fr \not\models_{\mathcal{PAR}_m} \delta$ for some $\delta \in prem(V(\alpha))$ or $Fr \models_{\mathcal{PAR}_m} cons(V(\alpha))$. Thus $Fr \models_{\mathcal{PAR}_m} V(\alpha)$ and hence $Fr \models_{\mathcal{PAR}_V} \alpha$. $\square$

The well known result of existence of a free model for positive conditional specifications can be obtained just as a corollary of theorem 2.4.5.

Cor. 2.4.6 If PSp is a positive conditional specification, then for all families X of variables $(Fr(PSp, X), m(PSp, X))$ is free for X in $PMod(PSp)$.

Proof. Let Fr denote $Fr(PSp, X)$ and m denote $m(PSp, X)$. Because of theorem 2.4.5, it is sufficient to show that SNF(PSp,X) is empty. Assume that ϕ satisfies snf_1 and snf_2 and show that ϕ does not satisfy snf_3. Because of snf_1, ϕ is $\alpha[t_y/y \mid y \in Var(\alpha)]$ for some $\alpha \in Ax$ and some $t_y \in T_\Sigma(X)$ s.t. $Fr \models_{\mathcal{PAR}_m} D(t_y)$; since PSp is a positive conditional specification, all the premises of α are existential equalities. Thus $Fr \models_{\mathcal{PAR}_m} \delta$ implies $A \models_{\mathcal{PAR}_V} \delta$ for all models A, all valuations $V: X \to A$ and all $\delta \in prem(\phi)$, by definition of Fr. Analogously $Fr \models_{\mathcal{PAR}_m} D(t_y)$ implies $A \models_{\mathcal{PAR}_V} D(t_y)$ and hence $V': Var(\alpha) \to A$, defined by $V'(y) = {t_y}^{A,V}$, is a valuation; moreover, since A is a model of sp, $A \models_{\mathcal{PAR}_{V'}} \alpha$ and hence $A \models_{\mathcal{PAR}_V} \phi$. Therefore from $A \models_{\mathcal{PAR}_V} \phi$ and $A \models_{\mathcal{PAR}_V} \delta$ for all $\delta \in prem(\phi)$, $A \models_{\mathcal{PAR}_V} cons(\phi)$ follows for all models A and all valuations $V: X \to A$, so that $Fr \models_{\mathcal{PAR}_m} cons(\phi)$, i.e. snf_3 does not hold. □

Note that there exist classes $PMod(sp)$ admitting free objects for all families X of variables which are not definable by only positive conditional formulas, as the following example shows.

spec sp_3 =
  sorts s
  opns
    $a, b, c, d: \to s$
  axioms
    $\alpha \quad a{=}b \supset c{=}d$

Then for all families X of variables there exists a free object (Fr, m) for sp_3, defined by:

Algebra Fr =
  $s^{Fr} = X$
  $a^{Fr}, b^{Fr}, c^{Fr}, d^{Fr}$ are undefined
  $m(x) = x$

In order to show that $PMod(sp_3)$ cannot be the model class of a positive conditional specification it is sufficient to show that it is not closed under non-empty products, being the model class of a positive conditional specification a quasi-variety (see e.g. [92]). Let A and B be the models of sp_3 defined by:

Algebra A =
  $s^A = \{1, 2\}$
  $a^A = 1, b^A$ undefined
  $c^A = 2, d^A$ undefined

Algebra B =
$s^B = s^A$
a^B undefined, $b^B = 1$
$c^B = 2, d^B$ undefined

Both are models of sp_3; but their product is the algebra C defined by:

Algebra C =
$s^C = \{1,2\} \times \{1,2\}$
a^C, b^C, d^C undefined, $c^C = (2,2)$

that is not a model of sp_3. □

Both in the total conditional and in the partial positive conditional cases the closure under non-empty products of the model class guarantees that for any sort either the corresponding carrier is a singleton set in all models or there are models having this carrier of arbitrary cardinality. Indeed assume that there is a model A s.t. s^A has cardinality at least 2; then for any (possibly infinite) set I the product $\Pi_I A$ is a model, because the model class is closed under non-empty products, and $s^{\Pi_I A}$ has cardinality at least 2^I. The lack of closure under non-empty products for the partial conditional case makes this property false; more precisely, denoting by $\mid X \mid$ the cardinality of any set X, for any $n \in \mathbb{N}$ there exists a conditional specification $sp^n = (\Sigma^n, Ax^n)$ s.t.

- $\mid s^A \mid \leq n$ for all $A \in PMod(sp^n)$ and all $s \in S$;
- there exists $A \in PMod(sp^n)$ and $s \in S$ s.t. $\mid s^A \mid = n$.

Consider the following example.

spec Sp^n =
sorts s_i for $i = 1 \ldots n$
opns
$a_{i,j} \colon \to s_i$ for $i = 1 \ldots n, j = 1 \ldots i$
$\xi_i \colon s_{i+1} \to s_i$ for $i = 1 \ldots n-1$
axioms
$\alpha \quad x = a_{1,1}$
$\beta_{i,j} \quad D(a_{i,j})$ for $i = 1 \ldots n, j = 1 \ldots i$
$\gamma_i \quad \xi_i(x) = \xi_i(y) \supset x = y$ for $i = 1 \ldots n-1$

Let A be a model of sp^n and inductively show that $\mid {s_i}^A \mid \leq i$ for $i = 1 \ldots n$.

- ${s_1}^A$ is the singleton set $\{{a_{1,1}}^A\}$, because of α;

- assume that $\mid s_i{}^A \mid \leq i$; for any $a \in s_{i+1}{}^A$ either $\xi_i{}^A(a) \in s_i{}^A$ or $\xi_i{}^A(a)$ is undefined and hence there are $\mid s_i{}^A \mid +1$ possibilities to define $\xi_i{}^A(a)$; thus, as $\xi_i{}^A$ is an injective partial function by γ_i, $\mid s_{i+1}{}^A \mid \leq \mid s_i{}^A \mid +1 \leq i+1$.

Moreover it is easy to check that I is a model of sp^n, where I is defined by:

Algebra I =
$s_i{}^I = \{1 \dots i\}$
$a_{i,j}{}^I = j$ for all $i = 1 \dots n$ and all $j = 1 \dots i$
$\xi_i{}^I(j) = j$ for $j = 1 \dots i; \xi_i{}^I(i+1)$ is undefined for $i = 1 \dots n-1$

Note that in the total frame, from $\mid s_1{}^A \mid = 1$ and the injectivity of $\xi_i{}^A$, $\mid s_2{}^A \mid = 1$ follows and so, inductively, $\mid s_i{}^A \mid = 1$ for all i. On the converse, in the partial positive frame, changing γ_i in $\xi_i(x) =_e \xi_i(y) \supset x = y$, many different elements are allowed to have undefined image along ξ_i and hence there are models having the carriers of sort s_i of arbitrary cardinality for all $i = 1, \dots n$. □

Prop. 2.4.7 The existence of free objects for finitary conditional specifications is not decidable.

Proof. For every Thue system E over an alphabet A and every couple of non-empty strings u and w over A a specification $sp_{E,u,w}$ is exhibited s.t. $\mathrm{SNF}(sp_{\mathrm{E,u,w}}, \emptyset)$ is empty iff $u=w$ follows from E.

Therefore, since the set $\{(E, u, w) \mid E \vdash u=w\}$ is not decidable (see e.g. [20]) and the emptiness of $\mathrm{SNF}(\mathrm{Sp}_{\mathrm{E,u,w}}, \emptyset)$ is equivalent to the existence of an initial model for $Sp_{E,u,w}$, the existence of the initial model for the class of the conditional specifications $Sp_{E,u,w}$ is not decidable too.

It is well known that every Thue system E over an alphabet A may be represented by the total equational one-sorted specification

$$Sp_{E,A} = (\Sigma_A, E \cup \{\cdot(\cdot(x, y), z) = \cdot(x, \cdot(y, z)\})$$

where Σ_A consists of just one sort s, of a constant symbol $\underline{a}$ for each $a \in A$ and of a binary symbol $\cdot$ representing the concatenation, in the sense that for all non-empty streams u and w over A the equality $u = w$ follows from E iff it holds in all models of $Sp_{E,A}$.

Then for each Thue system E over A and all non-empty streams u and w over A let $Sp_{E,u,w}$ be the specification having the signature $\Sigma_A \cup (\{s'\}, \{b, b' \colon \rightarrow s'\})$ and the axioms

$$E \cup \{\cdot(\cdot(x, y), z) = \cdot(x, \cdot(y, z)\} \cup \{D(\cdot(x, y)), D(\underline{a}) \mid a \in A, u=w \supset D(b), b=b' \supset D(b)\}.$$

It is easy to check that $\mathrm{SNF}(\mathrm{Sp}_{\mathrm{E,u,w}}, \emptyset)$ is empty iff $u=w$ follows from E. □

2.4.2 Free objects and logical deduction

In the following when referring to generic formulas and inference systems formulas and inference systems within an infinitary logic which extends first-order logic by admitting denumerable conjunctions (, disjunctions) and quantification over denumerable sets of variables are considered (see e.g. [51]).

Notation. Since formulas of the form $\{D(x) \mid x \in X\} \cup \Theta \supset \epsilon$ will be often needed, a short notation for these formulas is introduced. In the following for all families $X = \{X_s\}_{s \in S}$ of variables the set of formulas $\{D(x) \mid x \in X_s, s \in S\}$ will be denote by $D(X)$.

Note that, as usual, quantification is always implicit and is universal, as it may be easily deduced from the definition of validity, i.e. every formula ϕ is a short notation for the formula $\{\forall x{:}\, s \mid x \in Var(\phi)_s\}_{s \in S}.\phi$. However (as in the total many-sorted frame) this short notation may cause a subtle error whenever empty carriers are allowed, as the following example shows.

spec sp_4 =
 sorts s_1, s_2
 opns
  $a, b{:} \to s_1$
  $f{:}\, s_2 \to s_1$
 axioms
  α_1 $D(a)$
  α_2 $D(b)$
  α_3 $a{=}f(x)$
  α_4 $f(x){=}b$

The deduction of $a{=}b$ from the axioms $a{=}f(x)$ and $f(x){=}b$ by transitivity is unsound; for example T_Σ is a model of sp_4 (actually it is initial) but $T_\Sigma \not\models_{\mathcal{PAR}} a{=}b$. This may happen, since $T_{\Sigma|s_2} = \emptyset$. □

Indeed Huet noted that, in the framework of many-sorted algebras, the family

$$\mathcal{R} = \{(t, t') \mid t, t' \in T_\Sigma(X)_{|s}, A{\models_{\mathcal{MS}}} t{=}t'\}_{s \in S}$$

may fail to be a congruence; in the example $T_\Sigma {\models_{\mathcal{MS}}} a{=}f(x)$ and $T_\Sigma {\models_{\mathcal{MS}}} f(x){=}b$, since $T_{\Sigma|s_2} = \emptyset$ and hence there does not exist any valuation for $\{x\}$ in T_Σ, but $T_\Sigma \not\models_{\mathcal{MS}} a{=}b$ so that $\mathcal{R}$ is not transitive. He suggested to avoid unsound ground deductions in the case of total algebras by restricting signatures to those whose corresponding carriers either are guaranteed to be non-empty by the existence of ground terms of that sort, or are in a sense absolutely disconnected by the non-empty carriers (the rigorous notion is that of sensible signature, see e.g. [50]). This approach fails in the partial framework since a ground term may be undefined in an algebra and hence its existence does not guarantee that the corresponding carrier is not empty; thus conditions on the signature are not sufficient to guarantee that all carriers are not empty.

The same problem was also tackled by Goguen and Meseguer in [64] with a particular interest to logical deduction. They proposed a system working on equalities of the form $(\forall X)t{=}t'$, where $Var(t) \cup Var(t') \subseteq X$, which produces $(\forall X - \{x\})t{=}t'$, eliminating a variable x from X, only if x does not appear in $t{=}t'$ and can be instantiated by a ground term. In this framework in the previous example from $(\forall \{x\})f(x) = b$ and $(\forall \{x\})a = f(x)$ it can be deduced $(\forall \{x\})a = b$, that holds also in T_Σ, but $a = b$ cannot be deduced, as x cannot be instantiated on a ground term, because $T_{\Sigma|s_2}$ is empty. A similar approach can be used also in the partial framework, only permitting the elimination of those variables that can be instantiated on ground terms whose definedness is provable. For another system of equational deduction handling the empty carrier problem see [58].

However the problem can be handled in a way that is more natural for the partial approach; indeed the existential equalities $t{=}_e t$, or definedness assertions $D(t)$, are at hand and can be used to replace in a formula the explicit indication of the variables to which the valuation refers. Thus the ([64])-like formula $(\forall X)\Delta \supset \epsilon$ here becomes $D(X) \cup \Delta \supset \epsilon$. Moreover, since $D(y)$ holds in all algebras for any y, the presence of $D(y)$ in the left-hand side of a conditional formula has the only effect of possibly increasing the set of variables appearing in the axiom and to which the valuation refers and hence the explicit indication of the definedness of the variables appearing in $\Delta \supset \epsilon$ can be forgot, what makes the partial deduction more concise.

In order to stress that $D(x)$ in the premises of a deduced formula just states the use af the variable x to make the deduction, a short notation is introduced for every inference system L, every conditional formula ϕ and every family X of variables; in the sequel the notation $L \vdash_X \phi$ stands for "$X' \subseteq X$ exists s.t. $L \vdash D(X') \wedge prem(\phi) \supset cons(\phi)$". Using the above remark, a general definition of logical systems which takes care of the empty-carrier problem can be given as follows.

Def. 2.4.8 For a conditional specification $sp = (\Sigma, Ax)$, a *conditional system* $L(sp)$ for sp, in the following simply called system if there is no ambiguity, is an inference system $L(sp)$ s.t.:

definedness of variables $L(sp) \vdash D(x)$ for all variables x;

axioms $L(sp) \vdash \alpha$ for all $\alpha \in Ax$;

congruence for all families X of variables the family $\equiv^{L(sp)}(X)$ consists of $\{\equiv^{L(sp)}(X)_s\}_{s\in S}$, where $\equiv^{L(sp)}(X)_s$ is the set
$\{(t,t') \mid t,t' \in T_\Sigma(X)_{|s}, L(sp)\vdash_X D(t), L(sp)\vdash_X t{=}t'\}$
is a congruence over $T_\Sigma(X)$ s.t.

$$Dom(\equiv^{L(sp)}(X)_s) = \{t \mid t \in T_\Sigma(X)_{|s}, L(sp)\vdash_X D(t)\};$$

substitution for all conditional formulas $\Delta \supset \eta$, all families $X \subseteq Var(\Delta \supset \eta)$, Z_x of variables s.t. $L(sp)\vdash_{Z_x} D(t_x)$ for all $x \in X$, $L(sp) \vdash \Delta \supset \eta$ implies

$$L(sp)\vdash_{\cup_{x\in X} Z_x}\{\delta[t_x/x \mid x \in X] \mid \delta \in \Delta\} \supset \eta[t_x/x \mid x \in X];$$

modus ponens for any countable set of elementary formulas Θ, Γ, Θ_γ and any elementary formula ϵ $L(sp) \vdash \Theta \cup \Gamma \supset \epsilon$ and $L(sp) \vdash \Theta_\gamma \supset \gamma$ for all $\gamma \in \Gamma$ implies

$$L(sp)\vdash_{\cup_{\gamma\in\Gamma} Var(\gamma)}\Theta \cup (\cup_{\gamma\in\Gamma}\Theta_\gamma) \supset \epsilon;$$

soundness for any formula ϕ, $L(sp) \vdash \phi$ implies $M \models_{\mathcal{PAR}} \phi$ for all $M \in PMod(sp)$. □

Note that, since $\equiv^{L(sp)}(X)$ is a congruence, in the conditional system $L(sp)$ the standard rules for reflexivity, symmetry, transitivity and functionality (in the slightly generalized form admitting a $D(X)$ in the premises) should be derivable.

In order to make the presentation more concrete and to prepare the way to a completeness result, a particular system is introduced, for the moment just as an example, is reminiscent of systems found in the literature (see e.g. [22, 21]), but taking care of the soundness problem like the previous remark suggests.

Def. 2.4.9 The $US(sp)$ system for a conditional specification $sp = (\Sigma, Ax)$ consists of the axioms Ax and of the following axioms and inference rules, where, as usual, $\epsilon \in \mathrm{EForm}(\Sigma, \mathit{Var})$, $\Delta, \Delta_\gamma, \Gamma$ are countable subsets of $\mathrm{EForm}(\Sigma, \mathit{Var})$, $x \in \mathit{Var}$ and $t, t', t'', t_i, t'_i \in T_\Sigma(\mathit{Var})$.

1. *Definedness of variables*
   $D(x)$

2. *Congruence*

   (a) $t{=}t$

   (b) $t{=}t' \supset t'{=}t$

(c) $t{=}t' \wedge t'{=}t'' \supset t{=}t''$

(d) $t_1{=}t'_1 \wedge \ldots \wedge t_n{=}t'_n \supset op(t_1, \ldots, t_n){=}op(t'_1, \ldots, t'_n)$

3. *Strictness*
$D(op(t_1, \ldots, t_n)) \supset D(t_i)$

4. *Definedness and equality*
$D(t) \wedge t{=}t' \supset D(t')$

5. *Modus Ponens*

$$\frac{\Delta \cup \Gamma \supset \epsilon, \{\Delta_\gamma \supset \gamma \mid \gamma \in \Gamma\}}{D(Var(\Gamma) - Var(\cup_{\gamma\in\Gamma}\Delta_\gamma \cup \Delta \supset \epsilon)) \cup \Delta \cup (\cup_{\gamma\in\Gamma}\Delta_\gamma) \supset \epsilon}$$

6. *Instantiation/Abstraction*

$$\frac{\Delta \supset \epsilon}{\{D(t_x) \mid x \in X_s, s \in S\} \cup \{\delta^\sharp \mid \delta \in \Delta\} \supset \epsilon^\sharp}$$

where $\eta^\sharp$ denotes $\eta[t_x/x \mid x \in X_s, s \in S]$ for every elementary formula η and $t_x \in T_\Sigma(Var)_{|s}$ for all $x \in X_s$. □

Remark. It is worth to note that instantiation and abstraction are both handled by the above rule 6 to keep the system as economical as possible; indeed instantiation corresponds to X being the family of variables of the formula that has to be instantiated and abstraction corresponds to X being a family of variables which do not appear in the formula that has to be abstracted and $t_x = x$ for all $x \in X$. Thus rule 6 may be replaced by the following $*$ and $\star$.

$*$ *Instantiation*

$$\frac{\Delta \supset \epsilon}{\{D(t_x) \mid x \in X_s, s \in S\} \cup \{\delta^\sharp \mid \delta \in \Delta\} \supset \epsilon^\sharp}$$

where $\eta^\sharp$ denotes $\eta[t_x/x \mid x \in Var(\Delta \supset \epsilon)_s, s \in S]$ for every elementary formula η and $t_x \in T_\Sigma(Var)_{|s} \forall x \in X_s$

$\star$ *Abstraction*

$$\frac{\Delta \supset \epsilon}{D(X) \cup \Delta \supset \epsilon}$$

Obviously both $*$ and $\star$ are a particular case of 6 and it is easy to check that any application of 6 may be replaced by an application of $\star$ to increase the number of variables and an application of $*$ to instantiate the variables:

Using $\star$ and $*$

$$\frac{\Delta \supset \epsilon}{D(X) \cup \Delta \supset \epsilon}$$

$$\frac{D(X) \cup \Delta \supset \epsilon}{\Gamma_{Def} \cup \{\delta[t_x/x \mid x \in X_s, s \in S] \mid \delta \in \Delta\} \supset \epsilon[t_x/x \mid x \in X_s, s \in S]}$$

Using 6

$$\frac{\Delta \supset \epsilon}{\Gamma_{Def} \cup \{\delta[t_x/x \mid x \in X_s, s \in S] \mid \delta \in \Delta\} \supset \epsilon[t_x/x \mid x \in X_s, s \in S]}$$

where $\Gamma_{Def} = \{D(t_x) \mid x \in X_s, s \in S\}$. □

Prop. 2.4.10 For all conditional specifications sp, $US(sp)$ is a system for sp.

Proof. Because of rule 1, the condition on definedness of variables is satisfied; since the axioms of sp belong to $US(sp)$, obviously the condition on the axioms is satisfied and, because of rules 2… 5, also the condition on congruence is satisfied. Moreover, because of rules 5 and 6, $US(sp)$ satisfies the condition on substitution and, because of rule 5, the condition on modus ponens. Thus it is sufficient to show that it is sound; this is done by induction over the rules of $US(sp)$. It is obvious that the rules 1… 4 are sound, by definition of validity for the definedness predicate and the equality; thus only consider rules 5 and 6.

Assume that the hypotheses of rule 5 are satisfied, i.e. that $US(sp) \vdash \phi$, where ϕ is $\Delta \cup \Gamma \supset \epsilon$, and that $US(sp) \vdash \phi_\gamma$ for all $\gamma \in \Gamma$, where ϕ_γ is $\Delta_\gamma \supset \gamma$; then $US(sp) \vdash \phi'$, where ϕ' is

$$D(Var(\Gamma) - Var(\Delta \cup \cup_{\gamma\in\Gamma}\Delta_\gamma \supset \epsilon)) \cup \Delta \cup (\cup_{\gamma\in\Gamma}\Delta_\gamma) \supset \epsilon,$$

and show that $A \models_{\mathcal{PAR}} \phi$ and $A \models_{\mathcal{PAR}} \phi_\gamma$ for all $\gamma \in \Gamma$ implies $A \models_{\mathcal{PAR}} \phi'$ for all $A \in PMod(sp)$. Then let V be a valuation for $Var(\phi')$ in $A \in PMod(sp)$ s.t. $A \models_{\mathcal{PAR}_V} \delta$ for all $\delta \in prem(\phi')$ and show that $A \models_{\mathcal{PAR}_V} cons(\phi')$.

First of all note that $Var(\phi')$ is the same set as $Var(\phi) \cup \cup_{\gamma\in\Gamma} Var(\phi_\gamma)$, so that V is also a valuation for $Var(\phi)$ and for $Var(\phi_\gamma)$ in A. Moreover, for all $\gamma \in \Gamma$, $prem(\phi_\gamma) \subseteq prem(\phi')$ and hence, because of $A \models_{\mathcal{PAR}_V} \delta$ for all $\delta \in prem(\phi')$, $A \models_{\mathcal{PAR}_V} \delta$ for all $\delta \in prem(\phi_\gamma)$; thus, since by inductive hypothesis $A \models_{\mathcal{PAR}} \phi_\gamma$, $A \models_{\mathcal{PAR}_V} \gamma$ for all $\gamma \in \Gamma$.

Analogously, since $\Delta \subseteq prem(\phi')$, $A \models_{\mathcal{PAR}_V} \delta$ for all $\delta \in \Delta$ and hence $A \models_{\mathcal{PAR}_V} \delta$ for all $\delta \in prem(\phi) = \Gamma \cup \Delta$, so that $A \models_{\mathcal{PAR}_V} cons(\phi)$, because $A \models_{\mathcal{PAR}_V} \phi$. Finally $cons(\phi) = cons(\phi')$ and hence $A \models_{\mathcal{PAR}_V} \phi'$.

The case of rule 6 can be similarly developed. Assume that $US(sp) \vdash \phi$, where ϕ is $\Delta \supset \epsilon$; then $US(sp) \vdash \phi'$, where ϕ' is the formula

$$\{D(t_x) \mid x \in X_s, s \in S\} \cup \{\delta[t_x/x \mid x \in X_s, s \in S] \mid \delta \in \Delta\} \supset \epsilon[t_x/x \mid x \in X_s, s \in S],$$

and show that $A \models_{\mathcal{PAR}} \phi$ implies $A \models_{\mathcal{PAR}} \phi'$ for all $A \in PMod(sp)$.

Then let V be a valuation for $Var(\phi')$ in $A \in PMod(sp)$ s.t. $A \models_{\mathcal{PAR}_V} \delta$ for all $\delta \in prem(\phi')$ and show that $A \models_{\mathcal{PAR}_V} cons(\phi')$.
Since $\{D(t_x) \mid x \in X_s, s \in S\} \subseteq prem(\phi')$ and $A \models_{\mathcal{PAR}_V} \delta$ for all $\delta \in prem(\phi')$, then $A \models_{\mathcal{PAR}_V} D(t_x)$ for all $x \in X$; thus $V'\colon Var(\phi) \to A$ defined by $V'(x) = {t_x}^{A,V}$ if $x \in X$, otherwise $V'(x) = V(x)$ is a valuation for $Var(\phi)$ in A. Moreover, by definition of V', for all elementary formulas δ on $Var(\phi)$ $A \models_{\mathcal{PAR}_{V'}} \delta$ iff $A \models_{\mathcal{PAR}_V} \delta[t_x/x \mid x \in X_s, s \in S]$. Thus, since $A \models_{\mathcal{PAR}_V} \delta[t_x/x \mid x \in X_s, s \in S]$ for all $\delta \in prem(\phi)$, $A \models_{\mathcal{PAR}_{V'}} \delta$ follows, for all $\delta \in prem(\phi)$, and hence, since $A \models_{\mathcal{PAR}} \phi$ too, $A \models_{\mathcal{PAR}_{V'}} cons(\phi)$, i.e. $A \models_{\mathcal{PAR}_{V'}} \epsilon$, so that $A \models_{\mathcal{PAR}_V} \epsilon[t_x/x \mid x \in X_s, s \in S]$, i.e. $A \models_{\mathcal{PAR}_V} cons(\phi')$. $\square$

The focus of algebraic logic deduction is on *equational* deduction, because an inference system complete w.r.t. the (existential) equations gives the (initial) free model, if any. Since equations with explicit quantification $\forall(X \cup Var(\epsilon)).\epsilon$ are here equivalent to formulas of the form $D(X) \supset \epsilon$, notions of soundness and completeness also dealing with such particular conditional formulas are given; these notions subsume the usual ones only dealing with equalities.

Def. 2.4.11 Let sp be a conditional specification, X be a family of variables and $L(sp)$ be a system for sp. In the following $\mathsf{Fr}(L(sp), X)$ stands for $T_\Sigma(X)/\equiv^{L(sp)}(X)$ and $\mathsf{m}(L(sp), X)\colon X \to \mathsf{Fr}(L(sp), X)$ is the valuation defined by $\mathsf{m}(L(sp), X)(x) = [x]$. Moreover let $\mathrm{EEq}(L(sp), X)$ be the following set

$$\{D(t) \mid t \in T_\Sigma(X)_{|s}\} \cup$$
$$\{t{=}t' \mid t, t' \in T_\Sigma(X)_{|s}, L(sp) \vdash_X D(t) \text{ or } L(sp) \vdash_X D(t')\}.$$

$L(sp)$ is *existentially equationally complete for X and sp*, in the following simply called eeq-complete, iff for any $\epsilon \in \mathrm{EEq}(L(sp), X)$ if $M \models_{\mathcal{PAR}} D(X) \supset \epsilon$ for all $M \in PMod(sp)$, then $L(sp) \vdash_X \epsilon$.

$L(sp)$ is *strongly equationally complete for X and sp*, in the following simply called seq-complete, iff for any elementary formula ϵ over Σ and X if $M \models_{\mathcal{PAR}} D(X) \supset \epsilon$ for all $M \in PMod(sp)$, then $L(sp) \vdash_X \epsilon$. $\square$

It is worth noting that the easier formulation of completeness
...if $M \models_{\mathcal{PAR}} D(X) \supset \epsilon$ for all $M \in PMod(sp)$, then $L(sp) \vdash D(X) \supset \epsilon$...
is too restrictive. Indeed, although most systems have a rule of abstraction which allows to deduce $L(sp) \vdash D(X') \cup \{x\} \supset \epsilon$ from $L(sp) \vdash D(X') \supset \epsilon$, in general if X is an infinite set $L(sp) \vdash D(X') \supset \epsilon$ does not imply $L(sp) \vdash D(X) \supset \epsilon$ and in particular this happens for any finitary system, as the one presented in the Sect. 4.4.1 below and in [7].

Note that $\mathsf{Fr}(L(sp), X)$ is well defined because of condition on congruence of Def. 2.4.8, which makes $\equiv^{L(sp)}(X)$ a congruence, and that $\mathsf{m}(L(sp), X)$ is really a valuation, i.e. a total function, because of condition on definedness of variables of Def. 2.4.8.

Any formula in $\mathrm{EEq}(L(sp), X)$ plays the role of a (quantified) existential equality; this justifies calling "existentially equational completeness" the completeness w.r.t. $\mathrm{EEq}(L(sp), X)$. Thus an eeq-complete system deduces all existential equalities holding in all models and hence $\mathsf{Fr}(L(sp), X)$ is exactly $Fr(sp, X)$.

Prop. 2.4.12 For all conditional systems $L(sp)$ for sp the system $L(sp)$ is eeq-complete for X and sp iff $\mathsf{Fr}(L(sp), X)$ coincides with $Fr(sp, X)$.
Proof. Let $K(sp, X)$ be shortly denoted by K and $\equiv^{L(sp)}(X)$ by $\equiv$. By definition $\mathsf{Fr}(L(sp), X)$ coincides with $Fr(sp, X)$ iff $\equiv$ and K are the same and hence it is sufficient to show that $L(sp)$ is eeq-complete iff $\equiv$ and K are the same.

Because of the soundness of $L(sp)$, $\equiv \subseteq K$. Indeed if $(t, t') \in \equiv$, then, by definition of $\equiv$, there exist $X', X'' \subseteq X$ s.t. $L(sp) \vdash D(X') \supset D(t)$ and $L(sp) \vdash D(X'') \supset t{=}t'$.
Thus, as $L(sp)$ is sound, for all models A and all valuations V for X in A, $A \models_{\mathcal{PAR}_V} D(X') \supset D(t)$ and $A \models_{\mathcal{PAR}_V} D(X'') \supset t{=}t'$; moreover, by definition of valuation, $A \models_{\mathcal{PAR}_V} D(x)$ for all $x \in X$ and hence $A \models_{\mathcal{PAR}_V} D(t)$, $A \models_{\mathcal{PAR}_V} t{=}t'$, i.e. $(t, t') \in K^{A,V}(X)$. Therefore $(t, t') \in K$.

So the thesis is $K \subseteq \equiv$ iff $L(sp)$ is eeq-complete.

- $\Rightarrow$ Let formulas in $\mathrm{EEq}(L(sp), X)$ be shortly denoted by existential equalities; assume that $t{=_e}t' \in \mathrm{EEq}(L(sp), X)$ and $L(sp) \not\vdash D(X') \supset t{=_e}t'$ for all $X' \subseteq X$. Since $L(sp) \not\vdash D(X') \supset t{=_e}t'$ for any $X' \subseteq X$, $(t, t') \notin \equiv$ and hence, as $\equiv = K$, $(t, t') \notin K$. Thus there exist a model $A \in PMod(sp)$ and a valuation $V: X \to A$ s.t. $t^{A,V} \neq t'^{A,V}$, i.e. $A \not\models_{\mathcal{PAR}_V} t{=_e}t'$.

- $\Leftarrow$ If $(t, t') \in K$, then for all models A and all valuations $V: X \to A$, $(t, t') \in K^{A,V}(X)$, i.e. both $A \models_{\mathcal{PAR}_V} D(t)$ and $A \models_{\mathcal{PAR}_V} t{=}t'$ and hence $A \models_{\mathcal{PAR}_V} D(X) \supset D(t)$ and $A \models_{\mathcal{PAR}_V} D(X) \supset t{=}t'$. Thus, as $L(sp)$ is eeq-complete, $L(sp) \vdash_X D(t)$ and $L(sp) \vdash_X t{=}t'$, i.e. $(t, t') \in \equiv$. $\Box$

Note that, as the following example shows, in general a conditional system is not eeq-complete for conditional specifications also in the more restrictive hypothesis that there exists a free object for every family of variables; moreover the free object may be different from $(\mathsf{Fr}(L(sp), X), \mathsf{m}(L(sp), X))$.

spec sp_5 =
  sorts s
  opns
    $a\colon \to s$
    $f, g\colon s \to s$
  axioms
    $\alpha_1 \quad f(x)=g(x) \supset D(f(a))$
    $\alpha_2 \quad D(f(x)) \supset f(x)=g(x)$
    $\alpha_3 \quad D(g(x)) \supset f(x)=g(x)$
    $\alpha_4 \quad D(a)$

Because of α_2 and α_3, $f(x)=g(x)$ holds in all models of sp_5 and hence, instantiating α_1 for $x = a$, that is defined by α_4, $D(f(a))$ holds too, while, for example, $US(sp_5) \not\vdash D(f(a))$ and hence $US(sp_5)$ is not eeq-complete for $\emptyset$ and sp_5. Moreover for every family X of variables $Fr(sp_5, X)$, defined as follows, is a model and hence, because of Theorem 2.4.5, it is the free object for X in $PMod(sp_5)$.

Algebra $Fr(sp_5, X)$ =
  $s^{Fr(sp_5,X)} = X \cup \{1, 2\}$
  $a^{Fr(sp_5,X)} = 1$
  $f^{Fr(sp_5,X)} = \phi = g^{Fr(sp_5,X)}$

where ϕ is defined only on 1 and $\phi(1) = 2$. □

Thus in general the existence of a free object does not imply that $\mathsf{Fr}(L(sp), X)$ is a model; however if $\mathsf{Fr}(L(sp), X)$ is a model, then it is also the free object for X in $PMod(sp)$, $L(sp)$ being sound, as the following proposition shows.

Prop. 2.4.13 For all families X of variables and all systems $L(sp)$ for sp the algebra $\mathsf{Fr}(L(sp), X)$ is a model of sp iff $(\mathsf{Fr}(L(sp), X), \mathsf{m}(L(sp), X))$ is free for X in $PMod(sp)$.

Proof. Let Fr denote $\mathsf{Fr}(L(sp), X)$ and m denote $\mathsf{m}(L(sp), X)$.

- ⇒ Because of Theorem 2.4.5, it is sufficient to show that $\equiv^{L(sp)}(X) = K(Sp, X)$. As in Prop. 2.4.12, $\equiv^{L(sp)}(X) \subseteq K(Sp, X)$, because of soundness of $L(sp)$. On the contrary if Fr is a model, then $K(sp, X) \subseteq K^{\mathsf{Fr},\mathsf{m}} = \equiv^{L(sp)}(X)$; thus $K(Sp, X) = \equiv^{L(sp)}(X)$.

$\Leftarrow$ Obvious. $\square$

It is now convenient to give a notion of naughty formula related to a system, since it allows to connect the existence of a free model with logical inference systems.

Def. 2.4.14 Let $L(sp)$ be a system for a conditional specification sp, X be a family of variables, Fr denote $\mathsf{Fr}(L(sp), X)$ and m denote $\mathsf{m}(L(sp), X)$. The set NF$(L(sp)$,X) (NF for Naughty Formulas) consists of all conditional formulas ϕ s.t.

nf_1 ϕ is $\alpha[t_y/y \mid y \in Var(\alpha)]$ for some $\alpha \in Ax$ and $t_y \in T_\Sigma(X)$ s.t. $\mathsf{Fr}\models_{\mathcal{PAR}_\mathsf{m}} D(t_y)$;

nf_2 $\mathsf{Fr}\models_{\mathcal{PAR}_\mathsf{m}}\delta$ for all $\delta \in prem(\phi)$;

nf_3 $\mathsf{Fr}\not\models_{\mathcal{PAR}_\mathsf{m}} cons(\phi)$. $\square$

Prop. 2.4.15 The set NF$(L(sp)$,X) consists of all conditional formulas ϕ s.t.

nf'_1 ϕ is $\alpha[t_y/y \mid y \in Var(\alpha)]$ for some $\alpha \in Ax$ and $t_y \in T_\Sigma(X)$ s.t. $L(sp)\vdash_X D(t_y)$;

nf'_2 for each $\delta \in prem(\phi)$ $L(sp)\vdash_X \delta$ or δ is $t{=}t'$ and $L(sp)\not\vdash_X D(t)$ and $L(sp)\not\vdash_X D(t')$;

nf'_3 $L(sp)\not\vdash_X cons(\phi)$ and $cons(\phi) \in$ EEq$(L(sp), X)$.

Proof. By definition of $\mathsf{Fr}(L(sp), X)$ and $\mathsf{m}(L(sp), X)$, for all terms $t, t' \in T_\Sigma(X)$:

- $\mathsf{Fr}(L(sp), X)\models_{\mathcal{PAR}_\mathsf{m}} D(t)$ iff $L(sp)\vdash_X D(t)$ and
- $\mathsf{Fr}(L(sp), X)\models_{\mathcal{PAR}_\mathsf{m}} t{=}t'$ iff $L(sp)\vdash_X t{=}t'$ or both $L(sp)\not\vdash_X D(t)$ and $L(sp)\not\vdash_X D(t')$.

Thus the equivalence between nf_i and nf'_i easily follows. $\square$

Theorem 2.4.16 Let $L(sp)$ be a system for sp and X be a family of variables. The set NF$(L(sp)$,X) is empty iff $\mathsf{Fr}(L(sp), X)$ is a model of sp.
Proof. Let Fr denote $\mathsf{Fr}(L(sp), X)$ and m denote $\mathsf{m}(L(sp), X)$.

$\Rightarrow$ Let α be an axiom of sp and V be a valuation for $Var(\alpha)$ in Fr. Then $t_{V,y} \in Dom(\equiv^{L(sp)}(X))$, i.e. $\mathsf{Fr} \models_{\mathcal{PAR}_\mathsf{m}} D(t_{V,y})$, for all $y \in Var(\alpha)$ and hence $V(\alpha)$ satisfies condition nf_1. Thus, since NF($L(sp)$, X) is empty, $V(\alpha)$ does not satisfy condition nf_2 or nf_3, i.e. $\mathsf{Fr} \models_{\mathcal{PAR}_\mathsf{m}} V(cons(\alpha))$ or $\mathsf{Fr} \not\models_{\mathcal{PAR}_\mathsf{m}} \delta$ for some $\delta \in V(prem(\alpha))$, i.e. , by definition of $V(\alpha)$, $\mathsf{Fr} \models_{\mathcal{PAR}_V} cons(\alpha)$ or $\mathsf{Fr} \not\models_{\mathcal{PAR}_V} \delta$ for some $\delta \in prem(\alpha)$ and hence $\mathsf{Fr} \models_{\mathcal{PAR}_V} \alpha$.

$\Leftarrow$ Let Fr belong to $PMod(sp)$ and ϕ satisfy conditions nf_1, nf_2 of Def. 2.4.14. Because of nf_1, ϕ is $\alpha[t_y/y \mid y \in Var(\alpha)]$ for some $\alpha \in Ax$ and $t_y \in T_\Sigma(X)$ s.t. $\mathsf{Fr} \models_{\mathcal{PAR}_\mathsf{m}} D(t_y)$ and hence the valuation V for $Var(\alpha)$ in Fr can be defined by $V(y) = [t_y]$. Since Fr is a model of sp, $\mathsf{Fr} \models_{\mathcal{PAR}_V} \alpha$, i.e., by definition of ϕ, $\mathsf{Fr} \models_{\mathcal{PAR}_\mathsf{m}} \phi$.

Therefore $\mathsf{Fr} \models_{\mathcal{PAR}_\mathsf{m}} \phi$, and, because of nf_2, that $\mathsf{Fr} \models_{\mathcal{PAR}_\mathsf{m}} \delta$ for all $\delta \in prem(\phi)$; thus $\mathsf{Fr} \models_{\mathcal{PAR}_\mathsf{m}} cons(\phi)$ and hence nf_3 does not hold. $\square$

Putting together Props. 2.4.12, and 2.4.13 and Theorem 2.4.16 the conditions for the existence and characterization of free models can be rephrased in terms of logical systems.

Theorem 2.4.17 Let sp be a conditional specification, X be a family of variables, Fr denote $\mathsf{Fr}(L(sp), X)$ and m denote $\mathsf{m}(L(sp), X)$. For every system $L(sp)$ for sp the following conditions are equivalent:

1. the set NF($L(sp)$, X) is empty;
2. the algebra Fr is a model of sp;
3. the couple $(\mathsf{Fr}, \mathsf{m})$ is free for X in $PMod(sp)$.

If (one of) the above conditions hold, then $L(sp)$ is eeq-complete for X and sp and $\mathsf{Fr}(L(sp), X) = Fr(sp, X)$.

If $L(sp)$ is eeq-complete then each one of the above conditions is equivalent to

4. there exists a free object for X in $PMod(sp)$;
5. there exists a free object for X in $PGen(sp, X)$.

Proof.

1 $\Leftrightarrow$ 2 By Theorem 2.4.16.

2 $\Leftrightarrow$ 3 By Prop. 2.4.13.

Assume that one among conditions 1, 2 and 3 holds, and show that $L(sp)$ is eeq-complete and that $\mathsf{Fr}(L(sp), X) = Fr(sp, X)$. Let $t =_e t'$ shortly denote a formula of $\mathrm{EEq}(L(sp), X)$ s.t. $L(sp) \not\vdash_X t =_e t'$; then, by definition of Fr and m, $\mathsf{Fr} \not\models_{\mathcal{PAR}_{\mathsf{m}}} t =_e t'$ and hence, as Fr is a model of sp, $L(sp)$ is eeq-complete. Thus, because of Prop. 2.4.12, $\mathsf{Fr}(L(sp), X) = Fr(sp, X)$.

Assume now that $L(sp)$ is eeq-complete.

4 $\Leftrightarrow$ 5 Because of Theorem 2.4.5

4 $\Leftrightarrow$ 2 Since $L(sp)$ is eeq-complete, $\mathsf{Fr} = Fr(sp, X)$ because of Prop. 2.4.12 and hence the thesis follows by Theorem 2.4.5. □

Both the eeq-completeness of every conditional system and the well known results guaranteeing the existence of a free model in the cases of positive conditional (see e.g. [22]) and total conditional (see e.g. [64]) specifications can be get as corollaries. At this end a preliminary result is needed.

Prop. 2.4.18 Let $sp = (\Sigma, Ax)$ be a conditional specification, X be a family of variables, $L(sp)$ be a conditional system for sp and $\alpha \in Ax$ be a positive conditional axiom. Any instantiation of α does not belong to $\mathrm{NF}(L(sp), \mathrm{X})$.

Proof. Assume that ϕ is an instantiation of α satisfying nf_1 and nf_2 and show that ϕ does not satisfy nf_3. Since ϕ is an instantiation of α satisfying nf_1, $\phi = \alpha[t_y/y \mid y \in Var(\alpha)]$ for some $t_y \in T_\Sigma(X)$ s.t. $L(sp) \vdash_X D(t_y)$ and hence, because of the condition on deducibility of the axioms and instantiation for conditional systems, $L(sp) \vdash_X \phi$, i.e. $L(sp) \vdash D(X') \cup prem(\phi) \supset cons(\phi)$ for some $X' \subseteq X$. Since α is a positive conditional axiom, by definition of $\mathsf{Fr}(L(sp), X)$, condition nf_2, i.e. $\mathsf{Fr}(L(sp), X) \models_{\mathcal{PAR}_{\mathsf{m}(L(sp),X)}} \delta$ for all $\delta \in prem(\phi)$, implies that $L(sp) \vdash_X \delta$, i.e. that $X_\delta \subseteq X$ exists s.t. $L(sp) \vdash D(X_\delta) \supset \delta$ for any $\delta \in prem(\phi)$.

Therefore, because of condition on modus ponens, by

$$L(sp) \vdash D(X') \cup prem(\phi) \supset cons(\phi)$$

and $L(sp) \vdash D(X_\delta) \supset \delta$ for all $\delta \in prem(\phi)$,

$$L(sp) \vdash D(X' \cup \cup_{\delta \in prem(\phi)} X_\delta \cup X'') \supset cons(\phi)$$

for some $X'' \subseteq Var(prem(\phi)) \subseteq X$, i.e. condition nf_3 does not hold. □

Cor. 2.4.19 Let X be a family of variables, $PSp = (\Sigma, Ax)$ be a positive conditional specification and $L(sp)$ be a conditional system for PSp.

1. $(\mathsf{Fr}(L(sp), X), \mathsf{m}(L(sp), X))$ is free for X in $PMod(PSp)$;

2. $L(sp)$ is eeq-complete for sp and X.

Proof. Because of Prop. 2.4.18, $\mathrm{NF}(L(sp), \mathrm{X})$ is empty; thus, because of Theorem 2.4.17, $(\mathsf{Fr}(L(sp), X), \mathsf{m}(L(sp), X))$ is free for X in $PMod(PSp)$ and hence, because of Theorem 2.4.17, $L(sp)$ is eeq-complete. □

Since total conditional specifications are a special case of (partial) positive conditional ones, this result applies to that frame too.

Remark. Note that in general a conditional system is not seq-complete for positive conditional specifications, as the following example shows.

spec sp_6 =
  sorts s
  opns
    $a\colon \rightarrow s$
    $f, g\colon s \rightarrow s$
  axioms
    α_1 $D(f(x)) \supset f(x)=g(x)$
    α_2 $D(g(x)) \supset f(x)=g(x)$

Since one among $D(f(x))$, $D(g(x))$ and $f(x)=g(x)$ holds by definition of strong equality, from α_1 and α_2 the validity of $f(x)=g(x)$ in all models of sp_6 follows, while, for example, $US(sp_6) \not\vdash f(x)=g(x)$ and hence $US(sp_6)$ is not seq-complete for $\emptyset$ and sp_6. □

2.5 Partial Higher-Order Specifications

As in [67], higher-order specifications are reduced to particular classes of first-order specifications.

Def. 2.5.1

- If S is a set of *basic sorts*, then the set $S^{\rightarrow}$ of *functional sorts* over S is inductively defined by: $S \subseteq S^{\rightarrow}$ and if $s_1, \ldots, s_n, s_{n+1} \in S^{\rightarrow}$, then $s = (s_1, \ldots, s_n \rightarrow s_{n+1}) \in S^{\rightarrow}$ for all $n \geq 1$.

  A subset $S' \subseteq S^{\rightarrow}$ is *downward-closed* iff $s_1, \ldots, s_n, s_{n+1} \in S'$ for all $(s_1, \ldots, s_n \rightarrow s_{n+1}) \in S'$.

- A *higher-order signature* $F\Sigma$ is a signature (S, F), where S is a downward-closed set of functional sorts, s.t. for any $s = (s_1, \ldots, s_n \rightarrow s_{n+1}) \in S$ with

$n \geq 1$ there exists a distinguished operator $\mathrm{apply}_s \in F_{ss_1,\ldots,s_n,s_{n+1}}$. In the sequel $\mathrm{apply}_s\,(f, a_1, \ldots, a_n)$ will be denoted by $f(a_1, \ldots, a_n)$ using an infix notation and dropping the sort indexes when there is no ambiguity. Moreover the apply functions will be not explicitly mentioned in the definitions of concrete functional signatures.

- Let $F\Sigma = (S, F)$ be a higher-order signature; then $A \in \mathrm{PA}(F\Sigma)$ is an *extensional partial algebra*, from now on *E-algebra*, iff it satisfies the following *extensionality condition*:
  for all $s = (s_1, \ldots, s_n \to s_{n+1}) \in S$, with $n \geq 1$ and for all $f, g \in s^A$, if for all $a_i \in {s_i}^A$, $i = 1, \ldots, n$, $f(a_1, \ldots, a_n) = g(a_1, \ldots, a_n)$, then $f = g$.

  The class of all E-algebras over $F\Sigma$ is denoted by $\mathrm{EPA}(F\Sigma)$.

- The institution $\mathcal{PHO} = (\mathbf{Sign}_{\mathcal{PHO}}, Sen_{\mathcal{PHO}}, Mod_{\mathcal{PHO}}, \models_{\mathcal{PAR}})$ of partial higher-order algebras consists of

  - $\mathbf{Sign}_{\mathcal{PHO}}$ is the subcategory of $\mathbf{Sign}_{\mathcal{PAR}}$ whose objects are higher-order signatures and whose morphisms $(\sigma, \phi)\colon F\Sigma \to F\Sigma'$ preserve functional sorts and apply functions, i.e. s.t.:

    * $\sigma(s_1, \ldots, s_n \to s_{n+1}) = \sigma(s_1) \ldots \sigma(s_n) \to \sigma(s_{n+1})$ for every functional sort $s_1, \ldots, s_n \to s_{n+1}$ of $F\Sigma$;
    * $\phi(\,\mathrm{apply}_s\,) = \mathrm{apply}_{\sigma(s)}$ for every functional sort s of $F\Sigma$.

    In the sequel let E denote the embedding of $\mathbf{Sign}_{\mathcal{PHO}}$ into $\mathbf{Sign}_{\mathcal{PAR}}$.

  - $Sen_{\mathcal{PHO}} = Sen_{\mathcal{PAR}} \circ E$;

  - $Mod_{\mathcal{PHO}}(F\Sigma)$ is the full subcategory of $Mod_{\mathcal{PAR}}(F\Sigma)$ whose objects are $\mathrm{EPA}(F\Sigma)$ for every higher-order signature $F\Sigma$ and $Mod_{\mathcal{PHO}}(\rho) = Mod_{\mathcal{PAR}}(\rho)_{|\mathrm{EPA}(F\Sigma')}$ for every $\rho\colon F\Sigma \to F\Sigma'$ in $\mathbf{Sign}_{\mathcal{PHO}}$.

- The institution $\mathcal{FPHO} = (\mathbf{Sign}_{\mathcal{PHO}}, Sen_{\mathcal{FPHO}}, Mod_{\mathcal{PHO}}, \models_{\mathcal{PAR}})$ of partial higher-order algebras with finitary sentences is the subinstitution of $\mathcal{PHO}$ whose sentences $Sen_{\mathcal{FPHO}}(F\Sigma)$ are $\{\phi \mid prem(\phi) \text{ are finite}\}$ for every higher-order signature $F\Sigma$.

- A *(positive) conditional higher-order specification* $(P)FSp = (F\Sigma, Ax)$ consists of a higher-order signature $F\Sigma$ and a set Ax of (positive) conditional axioms over $F\Sigma$. In general (positive) higher-order specifications will be denoted by $(P)FSp$. The class of *extensional models* of FSp, denoted by $\mathrm{EMod}(FSp)$, is $Mod(FSp) \cap \mathrm{EPA}(F\Sigma)$; while $\mathrm{EGMod}(FSp)$ is the class of extensional term-generated models, i.e. $GMod(FSp) \cap \mathrm{EPA}(F\Sigma)$. □

Note that for any higher-order signature $F\Sigma = (S, F)$, S is required to be downward closed in order that the operators apply_s have arity in $S^* \times S$.

Remark. Any $A \in \text{EPA}(F\Sigma)$ is isomorphic to an algebra where the carriers of higher-order sort $(s_1, \ldots, s_n \to s_{n+1})$ are subsets of the space of the partial functions from ${s_1}^A \times \ldots \times {s_n}^A$ into ${s_{n+1}}^A$ and the apply_s operators are interpreted in the standard way. Therefore in the following examples the higher-order carriers are assumed to be function spaces and the apply_s functions to be interpreted accordingly.

It is easy to check that first-order specifications are a special case of higher-order ones; indeed any (first-order) signature is a higher-order one, because there are no functional sorts and hence no apply function is required. Moreover the extensionality condition is satisfied by all partial algebras over a first-order signature, because all sorts are basic, so that also the models of a first-order specification and its *extensional* models coincide.

Note the difference between $f \in F_{\Lambda, s \to s'}$ and $f \in F_{s,s'}$; indeed in the first case the interpretation of f is an element of the corresponding carrier, and hence the extensionality condition (and any proper axiom involving a variable of sort $s \to s'$, too) applies to it, while in the second one the interpretation of f is a function (living in some meta-level world) from the carrier of sort s into the carrier of sort s'. This difference is better illustrated by an example.

spec $FSp_0 =$
- **sorts** $N, (N \to N)$
- **opns**
  - $Z\colon \to N$
  - $S, P, +2\colon N \to N \quad (*)$
  - $Inc\colon (N \to N) \to (N \to N)$
- **axioms**
  - $\alpha_1 \quad D(Z)$
  - $\alpha_2 \quad D(S(x))$
  - $\alpha_3 \quad P(S(x)) = x$
  - $\alpha_4 \quad +2(x) = S(S(x))$
  - $\alpha_5 \quad Inc(f)(x) = S(f(x))$
  - $\alpha_6 \quad f = Inc(g) \supset D(P(Z))$

Then there are no closed terms of sort $(N \to N)$ and the following algebra is obviously an extensional model of FSp_0, where $D(P(Z))$ does not hold:

Algebra $A =$
- $N^A = \mathbb{N}$
- $(N \to N)^A = \emptyset$

$Z^A = 0$
$S^A(x) = x + 1$ for all $x \in \mathbb{N}$
$P^A(x) = x - 1$ for all $x \geq 1$, $P^A(0)$ is undefined
$+2^A(x) = x + 2$ for all $x \in \mathbb{N}$
Inc^A is the totally undefined function

Consider now the specification FSp_0', equal to FSp_0 with the exception of the operations $(*)$ changed to

$$(\star) \quad S, P, +2\colon \to (N \to N);$$

It would seem natural that any model B of FSp_0 could be made a model of FSp_0', putting S^B, P^B and $+2^B$ into the carrier $(N \to N)^B$ and closing the carrier w.r.t. the Inc operation; but this cannot be done. Indeed for the specification FSp_0', S, P and $+2$ are closed terms of sort $(s \to s)$, defined in all models because of $\alpha_1, \alpha_2, \alpha_3$ and α_4; thus the variables f and g may be instantiated on S and $+2$. Moreover from α_5, for f instantiated on S, $Inc(S)(x) = S(S(x))$ follows and hence, from α_4, $Inc(S)(x) = +2(x)$. Thus, because of the extensionality condition, $Inc(S) = +2$ holds in all models and hence $D(P(Z))$ holds, from α_6.

Therefore $D(P(Z))$ holds in all extensional models of FSp_0' and hence there is no way of transforming the above algebra A into an extensional model of FSp_0' leaving the interpretations of P and Z unchanged. □

The above example shows that functions *between* carriers introduce less restrictions than constants *in* functional carriers. Note that the two levels describe conceptually different objects: functions between carriers are meta-objects which describe properties of the elements of the carriers, while the elements of functional carriers are the objects in discussion.

Since the class of extensional models is not closed under sub-objects, there is no guarantee that the initial (free) model (for X), if any, is term-generated (by X) nor that there exists an initial model in the whole model class iff there exists one in the subclass of the term-generated models.

Indeed both properties are missing for the class of extensional models, because there are specifications having a *non-term-generated initial model* and there are specifications whose term-generated model class has initial model while the whole model class has not.

Analogously to the case of higher-order *total* algebras, the class of extensional algebras is not closed w.r.t. subobjects; thus, in particular, the class of extensional algebras cannot be expressed as the model class of a conditional specification, because the model class of any conditional specification is closed under subobjects. But, while in the total case the extensional algebras are closed w.r.t. non-empty direct products (of course performed in the class of *all* algebras), as claimed for example by the theorem 5.3 in [62], in the partial frame also this closure is missing.

Fact 2.5.2 Let $F\Sigma$ be a higher-order signature; in general EPA($F\Sigma$) *is not closed w.r.t. subobjects, nor w.r.t. non-empty direct products*

Proof. Consider the signature $F\Sigma = (S, F)$, where $S = \{s, (s \to s)\}$ and $F = \{F_{w,s'}\}_{w \in S^*, s' \in S}$, with just two constants f and g of sort $s \to s$. Consider the algebras A , B and C, defined by:

Algebra $A =$

$$s^A = \{\bullet\}$$
$$(s \to s)^A = \{\perp, Id\}$$
$$f^A = Id \qquad\qquad g^A = \perp$$

$\perp(\bullet)$ is undefined, $Id(\bullet) = \bullet$

Algebra $B =$

$$s^B = s^A$$
$$(s \to s)^B = (s \to s)^A$$
$$f^B = \perp \qquad\qquad g^B = Id$$

Algebra $C =$

$$s^C = \emptyset$$
$$(s \to s)^C = (s \to s)^A$$
$$f^C = f^A \qquad\qquad g^C = g^A$$

Then obviously $A, B \in$ EPA($F\Sigma$), while $C \notin$ EPA($F\Sigma$) and C is a subalgebra of A, by definition. Therefore EPA($F\Sigma$) is not closed w.r.t. subobjects. Let $A \times B$ be defined as follows:

Algebra $A \times B =$

$$s^{A\times B} = \{(\bullet, \bullet)\}$$
$$(s \to s)^{A\times B} = (s \to s)^A \times (s \to s)^B$$
$$f^{A\times B} = (f^A, f^B) = (Id, \perp)$$
$$g^{A\times B} = (g^A, g^B) = (\perp, Id)$$

So $(s \to s)^{A\times B}$ has cardinality 4, while there are just two distinct partial functions from $s^{A\times B}$ into $s^{A\times B}$, the identity and the totally undefined function, because $s^{A\times B}$ has cardinality 1. Thus $A \times B \notin$ EPA($F\Sigma$) and hence EPA($F\Sigma$) is not closed w.r.t. non-empty direct products. □

As in the non-positive conditional partial case, the lack of closure w.r.t. non-empty products allows to define non-trivial specifications whose models are all of bounded cardinality.

It is easy to see that initial and terminal algebras in PA($F\Sigma$) are respectively characterized by $s^I = \emptyset$ for all $s \in S$, f^I totally undefined for all $f \in F$ and by $s^Z = \{\bullet\}$ for all $s \in S$, f^Z total for all $f \in F$. Obviously they are also extensional and hence initial and terminal in EPA($F\Sigma$); but, although EPA($F\Sigma$) has an initial model, in general both the class of extensional models and the class of term-generated models for equational specifications do not have initial model.

Fact 2.5.3 Let $F\Sigma = (S, F)$ be a higher-order signature and FSp be an equational specification $(F\Sigma, Ax)$. Then *in general there does not exist an E-algebra initial in* EMod(FSp) *nor in* EGMod(FSp).
Proof. Consider the following example.

spec FSp_1 =
  sorts $s, (s \to s)$
  constants
    e of type s
    f, g of type $s \to s$
  axioms
      $D(e)$
      $D(f)$
      $D(g)$

Proceed by contradiction assuming that there exists I initial in EMod(FSp_1) (resp. in EGMod(FSp_1)). Let F and G be the E-algebras defined by:

Algebra F =
  $s^F = \{\bullet\}$
  $(s \to s)^F = \{\perp, Id\}$
  $e^F = \bullet$
  $f^F = Id$
  $g^F = \perp$
 where $\perp(\bullet)$ is undefined, $Id(\bullet) = \bullet$

Algebra G =
  $s^G = s^F$
  $(s \to s)^G = (s \to s)^F$
  $e^G = \bullet$
  $f^G = \perp$
  $g^G = Id$

Both F and G belong obviously to EGMod(FSp_1); thus, because of the initiality of I, there exist two homomorphisms $p^F: I \to F$ and $p^G: I \to G$. It is routine to show that the existence of such p^F and p^G implies that for all $a \in s^I$ both $f^I(a)$ and $g^I(a)$ are undefined and hence that $f^I = g^I$, because of extensionality; thus $g^F = p^F(g^I) = p^F(f^I) = f^F$ follows, in contradiction with the definition of f^F and g^F. □

The above example suggests that for the existence of the initial model, the minimal *definedness* may conflict with the minimal *equality*. Indeed if the elements in the domain are too few, then the functions cannot be distinguished by them and hence the minimal definedness (on the arguments) may force the *maximal* equality (on the functions). For the same reason two functions having the same result over every tuple of terms because of the axioms, may differ on some *non-term-generated* argument-tuple, so that the equalities between ground terms holding in the term-generated models may be strictly more than the equalities holding in all models. In particular the equalities between ground terms holding in all the term-generated models may define an extensional algebra, so that there exists an initial model in EGMod(*FSp*), while the equalities between ground terms holding in all models are too few.

Fact 2.5.4 Let $F\Sigma = (S, F)$ be a higher-order signature and *FSp* be an equational specification $(F\Sigma, Ax)$ s.t. I is initial in EGMod(*FSp*). Then *in general I is not initial in* EMod(*FSp*) and the sets

$$\{\epsilon \mid \epsilon \in \mathrm{EForm}(F\Sigma, \emptyset), \mathrm{EMod}(FSp) \models_{\mathcal{PAR}} \epsilon\}$$

and

$$\{\epsilon \mid \epsilon \in \mathrm{EForm}(F\Sigma, \emptyset), \mathrm{EGMod}(FSp)) \models_{\mathcal{PAR}} \epsilon\}$$

are different.

Proof. Consider the following example.

> **spec** FSp_2 =
> > **sorts** $s_1, s_2, (s_1 \to s_2)$
> > **constants**
> > > e of type s_1
> > > f, g of type $(s_1 \to s_2)$
> >
> > **axioms**
> > > α_1 $D(f(e))$
> > > α_2 $f(e) = g(e)$

Then all term-generated models are isomorphic to I, defined by:

Algebra $I =$

$$s_1^I = \{\bullet\}$$
$$s_2^I = \{\bullet\}$$
$$(s_1 \to s_2)^I = \{Id\}$$
where $Id(\bullet) = \bullet$
$$e^I = \bullet$$
$$f^I = g^I = Id$$

So that I is initial in EGMod(FSp_2); however I is not initial in EMod(FSp_2), since there are (non term-generated) models A for which $f^A \neq g^A$.

Moreover EGMod(FSp_2)$\models_{\mathcal{PAR}} f = g$, while EMod($FSp_2$)$\not\models_{\mathcal{PAR}} f = g$, because $A \not\models_{\mathcal{PAR}} f = g$. □

In the total case if a family X of variables has a sufficiently high cardinality, then there exists the free model for X in the class of all extensional models of an equational specification (see theorems 3.7 and 5.7 of [62]). Instead in the partial case there are equational specifications whose classes of extensional models do not admit free models whatever the cardinality of the family of variables is.

Fact 2.5.5 Let $F\Sigma = (S, F)$ be a higher-order signature, FSp be an equational specification $(F\Sigma, Ax)$ and X be a family of variables of arbitrary cardinality. Then *in general there does not exist a free model for* X *in* EMod(FSp).

Proof. Consider again the specification FSp_1 and the algebras F and G defined in Fact 2.5.3 and show that there does not exist a free model for X in EMod(FSp_1).

Assume by contradiction that (Fr, v) is free in EMod(FSp_1) for a family X of variables. Let $V^F: X \to F$ and $V^G: X \to G$ be any valuations, which always exist, because F and G have all carriers non-empty. Because of the freeness of Fr, there exist two homomorphisms $p^F: \mathrm{Fr} \to F$ and $p^G: \mathrm{Fr} \to G$ s.t. $p^F \circ v = V^F$ and $p^G \circ v = V^G$. Thus, as in in Fact 2.5.3, $g^F = p^F(g^{\mathrm{Fr}}) = p^F(f^{\mathrm{Fr}}) = f^F$, in contradiction with the definition of f^F and g^F. □

Note that the above counter-example also applies to the subclass EGMod(FSp, X) of extensional models generated by the family X of variables, i.e.

$$\mathrm{EGMod}(FSp, X) = PGen(FSp, X) \cap \mathrm{EMod}(FSp),$$

because F and G, being term-generated, belong to EGMod(FSp, X).

In the (both total and partial) first-order case the free model (if any) for X in a class specified by conditional axioms is term-generated by X; the proof, in both cases, depends on the closure w.r.t. subalgebras.

In [62], under the hypothesis $T_{F\Sigma|s} \neq \emptyset$ for all $s \in S$, it is shown that if X has a sufficiently high cardinality, then a quotient of $T_{F\Sigma}(X)$ is a free model for X in the class of total models of a higher-order equational specification (but the same proof technique applies to the conditional case too). That result can be strengthened to show that if a free model for X exists, then it is term-generated by X, whatever the cardinality of X.

Theorem 2.5.6 Let $FSp = (F\Sigma, Ax)$ be a conditional higher-order specification s.t. $T_{F\Sigma_s} \neq \emptyset$ for all $s \in S$, denote by $\mathrm{TMod}(FSp)$ the class of total extensional models of FSp and assume that (Fr, v) is free for X in $\mathrm{TMod}(FSp)$. Then Fr is term-generated by X via v, i.e. $eval^{\mathrm{Fr},v}$ is surjective.

Proof. Because of theorems 3.7 and 5.7 of [62], there exists $Y \supseteq X$ of suitably high cardinality s.t. $(T_{F\Sigma}(Y)/\equiv, i)$ is free for Y in $\mathrm{TMod}(FSp)$, where $\equiv$ is the intersection of all kernels of natural evaluations $K^{A,V}$, and $i: Y \to T_{F\Sigma}(Y)/\equiv$ is the valuation defined by $i(y) = [y]$.

In the sequel let $[t]$ denote the equivalence class of any term t in $T_{F\Sigma}(Y)/\equiv$ and V denote any valuation $V: Y \to F$ s.t. $V(x) = v(x)$ for all $x \in X$ and $V(y) = {t_y}^F$ for some $t_y \in T_{F\Sigma}$ for all $y \in Y - X$. Note that there exists such a V because of the assumption $T_{F\Sigma|s} \neq \emptyset$, so that v may be extended to Y.

Two homomorphisms, $p_V: T_{F\Sigma}(Y)/\equiv \to \mathrm{Fr}$ and $h: \mathrm{Fr} \to T_{F\Sigma}(Y)/\equiv$ are exhibited s.t. $p_V([y]) = V(y)$ and $p_V \circ h = Id_{\mathrm{Fr}}$; then it is shown that this implies that $eval^{\mathrm{Fr},v}$ is surjective.

The existence of such p_V and h is now proved. Since $T_{F\Sigma}(Y)/\equiv \in TMod(FSp)$, $i \circ e: X \to T_{F\Sigma}(Y)/\equiv$ is a valuation, where e denotes the embedding of X into Y, and (Fr, v) is free for X in $\mathrm{TMod}(FSp)$, there exists a (unique) homomorphism $h: \mathrm{Fr} \to T_{F\Sigma}(Y)/\equiv$ s.t. $h \circ v = i \circ e$.

Since $\mathrm{Fr} \in TMod(FSp)$ and $(T_{F\Sigma}(Y)/\equiv, i)$ is free for Y in $\mathrm{TMod}(FSp)$, there exists a (unique) homomorphism $p_V: T_{F\Sigma}(Y)/\equiv \to \mathrm{Fr}$ s.t. $p_V \circ i = V$.

Since (Fr, v) is free for X in $\mathrm{TMod}(FSp)$, the unique homomorphism $p: \mathrm{Fr} \to \mathrm{Fr}$ s.t. $p \circ v = v$ is the identity, and hence $p_V \circ h = Id_{\mathrm{Fr}}$, because $p_V \circ h \circ v = p_V \circ i \circ e = V \circ e = v$.

The homomorphism $eval^{\mathrm{Fr},v}$ is proved to be surjective, i.e. for every $a \in s^{\mathrm{Fr}}$ a term $t \in T_{F\Sigma}(X)$ in exhibited s.t. $a = t^{\mathrm{Fr},v}$.

By definition of $T_{F\Sigma}(Y)/\equiv$, there exists $t' \in T_{F\Sigma}(Y)$ s.t. $h(a) = [t']$ and $[t'] = t'^{T_{F\Sigma}(Y)/\equiv,i}$. Thus $a = p_V(h(a)) = p_V(t'^{T_{F\Sigma}(Y)/\equiv,i})$; moreover $p_V(t'^{T_{F\Sigma}(Y)/\equiv,i}) = t'^{\mathrm{Fr},p_V \circ i}$ and, since $p_V \circ i = V$, $t'^{\mathrm{Fr},p_V \circ i} = t'^{\mathrm{Fr},V}$, so that $a = t'^{\mathrm{Fr},V}$.

Finally, by definition of V, $t'^{\mathrm{Fr},V} = t'[t_y/y | y \in Y - X]^{\mathrm{Fr},v}$; thus $a = t^{\mathrm{Fr},v}$ with $t = t'[t_y/y | y \in Y - X] \in T_{F\Sigma}(X)$ and hence $a \in eval^{\mathrm{Fr},v}(T_{F\Sigma}(X))$. $\square$

Therefore in the total case if (Fr, v) is free for X in the class of extensional models of a conditional specification, then it is also term-generated by X and hence it is the quotient of $T_{F\Sigma}(X)$ w.r.t. the intersection of the kernels of natural evaluations of $T_{F\Sigma}(X)$ in the models.

In the partial higher-order case both the closure under subalgebras (which is true for both the total and the partial first-order case) and the existence of free models for families of variables of sufficiently high cardinality (which is true for the total higher-order) are missing, so that neither style of proof applies. Indeed, rather surprisingly, there are positive conditional specifications whose initial model is not term-generated, as the following Fact 2.5.7 shows.

Fact 2.5.7 Let $F\Sigma = (S, F)$ be a higher-order signature and FSp be a positive conditional specification $(F\Sigma, Ax)$ s.t. $T_{F\Sigma_s} \neq \emptyset$ for all $s \in S$ and there exists an initial model I in $\mathrm{EMod}(FSp)$; then *in general I is not term-generated.*

Proof.

Let $\mathrm{MB}_{F\Sigma}$ denote the set of $F\Sigma$-equations which force the models of $(F\Sigma, \mathrm{MB}_{F\Sigma})$ to be the total trivial algebra, i.e. $\mathrm{MB}_{F\Sigma}$ consists of

$\{x_s = y_s | s \in S\}$ for some distinct $x_s, y_s \in Var_s$ and all $s \in S$;

$\{D_s(t) | t \in T_{F\Sigma}(X)_s\}$ where X is a family of denumerable sets of variables.

Then any *non-trivial* model of the set of axioms $\{\Delta \supset \epsilon \mid \epsilon \in \mathrm{MB}_{F\Sigma}\}$, from now on denoted by $\Delta \supset \mathrm{MB}_{F\Sigma}$, does not satisfy Δ.

Consider now the following specification $FSp_3 = (F\Sigma, Ax)$.

spec $FSp_3 =$

sorts $s, (s \to s), ((s \to s) \to s)$

constants

k of type s

$\perp$ of type $(s \to s)$

Θ_1, Θ_2 of type $((s \to s) \to s)$

opns

$\xi\colon (s \to s) \to ((s \to s) \to s)$

axioms

$x \in Var_s$; $f, g \in Var_{(s \to s)}$; $F \in Var_{(s \to s) \to s}$;

α $\quad D(\xi(f))$

α_1 $\quad x = k$

α_2 $\quad F(\perp) = k$

α_3 $\quad \Theta_1 = \Theta_2 \supset \mathrm{MB}_{F\Sigma}$

α_4 $\quad D(\perp(x)) \supset \mathrm{MB}_{F\Sigma}$

α_5 $\Theta_1(f) = k$
α_6 $\xi(\perp) = \Theta_1$
α_7 $\xi(f) = \xi(g) \supset f = g$

α_8 $D(k)$
α_9 $D(\perp)$
α_{10} $D(\Theta_1)$
α_{11} $D(\Theta_2)$

The specification FSp_3 has exactly one non-trivial model (modulo isomorphism), in the sequel called A, uniquely defined by the axioms:

- α_1 implies that s^A is a singleton set $\{1\}$ and $k^A = 1$; therefore, because of the extensionality, $(s \to s)^A$ has at most two elements: the identity Id and the totally undefined function $\perp$.

  If $(s \to s)^A$ is a singleton, then $(s \to s)^A = \{\perp^A\}$ and hence, because of α_2, $\Theta_1^A(\perp^A) = \Theta_2^A(\perp^A)$ would follow and hence $\Theta_1^A = \Theta_2^A$, by extensionality, contrary to α_3, which implies that ${\Theta_1}^A$ and ${\Theta_2}^A$ are different, because A is non-trivial. Therefore $(s \to s)^A = \{Id, \perp\}$.

- α_4 implies that $\perp^A = \perp$, because A is non trivial.

- α_2 implies that $F(\perp) = 1$ for any $F \in ((s \to s) \to s)^A$ and hence $((s \to s) \to s)^A$ has at most two elements: θ_1, defined by $\theta_1(Id) = 1$, $\theta_1(\perp) = 1$, and θ_2, defined by $\theta_2(\perp) = 1$, $\theta_2(Id)$ is undefined. Moreover $((s \to s) \to s)^A$ has at least two distinguished elements, Θ_1^A and Θ_2^A, because of α_3; therefore $((s \to s) \to s)^A = \{\theta_1, \theta_2\}$.

- α_5 implies that ${\Theta_1}^A = \theta_1$; thus, since ${\Theta_1}^A$ and ${\Theta_2}^A$ are different, ${\Theta_2}^A = \theta_2$.

- α_6 implies that $\xi^A(\perp) = \theta_1$.

- α and α_7 imply that ξ^A is a total injective function and hence $\xi^A(Id) = \theta_2$.

Thus the specification FSp_3 has the unique (extensional) non-trivial model A, defined by:

Algebra $A =$
$s^A = \{k^A\}$
$(s \to s)^A = \{\perp^A, Id\}$
$Id(k^A) = k^A$, $\perp^A(k^A)$ is undefined
$((s \to s) \to s)^A = \{\Theta_1^A, \Theta_2^A\}$
$\Theta_1^A(\perp^A) = k^A$, $\Theta_1^A(Id) = k^A$ and $\Theta_2^A(\perp^A) = k^A$, $\Theta_2^A(Id)$ is undefined
$\xi^A(\perp^A) = \Theta_1^A$ $\quad\quad$ $\xi^A(Id) = \Theta_2^A$

It is easy to check, then, that A is initial in EMod(FSp_3). Indeed there is just one homomorphism from A into the trivial model, because the trivial model is total and has just one element in any carrier. Moreover the only element in A which is not interpretation of a constant (and hence whose homomorphic image is not fixed *a priori*) is Id and the homomorphic image of Id must satisfy the equation $\Theta_2^A = h(\xi^A(Id)) = \xi^A(h(Id))$; since ξ^A is injective by α_7, $h(Id) = Id$ follows. Thus the identity is the only homomorphism of A into itself. $\square$

Remark. Note that in the above counter-example specification FSp_3 the only partial functions are the interpretations of the (implicit) apply operators that must be partial in order to specify carriers of partial functions. Moreover the equalities used to specify FSp_3 can be replaced by existential equalities leaving unaffected the model class. Thus FSp_3 is in some sense a *total* specification of partial functions and its pathologies only depend on the partiality of the objects that are being specified, i.e. the choice of the partial frame to axiomatize such a set of partial (higher-order) functions is immaterial.

Chapter 3

Relating Specification Formalisms

As illustrated by the last chapter for the particular case of the specification of partial functions, a considerable proliferation of formalisms has been produced in the last years, as a result both of theoretical investigations and of preliminary attempts at applications. One of the reasons behind this proliferation is that the convenience of a formalism may depend on the application and until now no "best formalism" has been found, though from time to time somebody claims that there is one.

Now this fragmentation of frames is conflicting with some essential requirements of any specification formalism, i.e. the ability of supporting modularity and refinement, which are clearly related to the problem of reuse of specifications. It is becoming important to assemble, possibly at different levels of implementative detail, specification modules in different formalisms; thus it is convenient (we believe fundamental) to have theoretically sound means for passing from one formalism to another in a way that preserves some essential properties. Rephrasing the title of a landmark paper by Burstall and Goguen [27], the issue is "putting together theories *from different formalisms* to make specifications".

Another hot point due to the existence of so many formalisms is the meaning of "equivalence" between approaches to the same problem in different frameworks. Indeed in the literature it is often claimed that a frame is *equivalent* to another one, usually in the sense that both solve the same kind of problems, or that in both the results are equivalently (un)satisfactory. But the meaning of equivalence is usually not formally defined and quite often used to denote different levels of relationship.

Both problems can be solved by a notion of formalism translator that on one hand allows theories expressed in different frameworks to be transformed in a common formalism and then assembled, and on the other relates the specifications defined in several formalisms, so that they can be compared. Since frameworks are formalized as institutions, such a translator should be some kind of morphism (in the categorical sense) between institutions. Starting from a significant set of concrete examples that should be captured by this notion of formalism translator, it is immediately apparent that the *morphisms of institutions*, originally defined by Burstall and Goguen in [44] to support categorical constructions on institutions, do not represent the wanted examples. Therefore a new kind of morphisms has been introduced in [2, 8], called *simulations*.

The basic idea of simulation is encoding the syntax, i.e. signatures and sentences, of a new frame by that of an already known formalism in a way consistent with the semantics, in order to transfer results and tools. To formalize the consistency of the translation of the syntax w.r.t. the semantics, every model of the new frame is required to be represented by at least one of the old frame that satisfies the same sentences (under translation). Thus a simulation consists of three components: the two maps translating new signatures and sentences into old ones, and a partial surjective map which translates old models into new ones.

Simulations are very close to *maps of institutions*, independenlty developped by Meseguer in [63], that differ from simulations because associate each new signature with an old theory, whose models alone are translated (so that the expressive power of the old institution is required to be sufficient to describe the domain of the model translator) non-necessary in a surjective way. For a more complete comparison of the two approches see Sect. 3.3.1 below

In this chapter simulations are introduced, compared with other morphisms of institutions and used to deeply analyze the different levels of relationship between logical formalisms, first following the total representations of partiality as a paradigmatic case to show that three degrees of connection can be found, respectively between individual models, categories of models and theories, and then applying the same technique to the coding of non-strict functions in total approaches, after a summary of the results in [4, 5] about non-strictness.

3.1 Simulations

To introduce the concept of simulation and the corresponding notation from an intuitive point of view, the reduction of many-sorted equational Horn-clauses logic $\mathcal{MS}$, to one-sorted logic $\mathcal{L}$, making explicit the typing of the variables (see e.g. [71]), is illustrated as a paradigmatic example of the kind of translations that have to be captured by the definition of simulation.

3.1.1 An Introductory Example

In the following a simulation is denoted by μ, possibly decorated.

Example 3.1.1 Let Σ be a many-sorted signature with sorts S and function symbols F. The translation of Σ into a one-sorted signature $\mu^M{}_{Sign}(\Sigma)$ is defined, by setting $\mu^M{}_{Sign}(\Sigma) = (Op', P')$, where Op'_n is the disjoint union of $F_{s_1 \ldots s_n, s}$, i.e. of the n-ary function symbols, disregarding the type of the arguments (so that any Σ-term is a (Op', P')-term, too) and P' contains only the typing predicates, i.e. $P'_1 = \{\_ : s \mid s \in S\}$, where the symbol $\_$ denotes the place of the argument in a prefix notation, and $P'_k = \emptyset$ for all $k \neq 1$.

With the help of the typing predicates, any many-sorted conditional equation over Σ can be translated into a one-sorted equivalent one over $\mu^M{}_{Sign}(\Sigma)$. Consider indeed a many-sorted formula

$$\xi = (V.t_1 = t'_1 \wedge \ldots \wedge t_n = t'_n \supset t = t')$$

over Σ and the variables x_i, where $V(x_i) = s_i$ for $i = 1 \ldots k$, and define

$$\mu^M{}_{Sen\,\Sigma}(\xi) = x_1 : s_1 \wedge \ldots \wedge x_k : s_k \wedge t_1 = t'_1 \wedge \ldots \wedge t_n = t'_n \supset t = t'.$$

Then in $\mu^M{}_{Sen\,\Sigma}(\xi)$, the translation of ξ over $\mu^M{}_{Sign}(\Sigma)$, the information about the typing of the variables is carried by the predicates $x_i : s_i$ in the premises.

To illustrate in which sense $\mu_{Sen\,\Sigma}(\xi)$ is equivalent to ξ, a class $dom(\mu)_\Sigma$ of one-sorted algebras is chosen, which soundly represents the many-sorted algebras and s.t. a one-sorted algebra satisfies $\mu_{Sen\,\Sigma}(\xi)$ iff the many-sorted algebra represented by it satisfies ξ. Again the typing predicates are used to simulate the different carriers of a many-sorted algebra: a one-sorted algebra A' is a sound representation of a many-sorted algebra A, denoted by $A = \mu_{Mod\,\Sigma}(A')$, iff whenever the arguments of a function are appropriately typed also the result is appropriately typed, i.e. $a_i : {s_i}^{A'}$ for $i = 1 \ldots n$ implies $f^{A'}(a_1, \ldots, a_n) : s^{A'}$ for any $f \in F_{s_1 \ldots s_n, s}$. If A' satisfies this condition, then A is the many-sorted algebra $(\{s^A\}_{s \in S}, \{f^A\}_{f \in F})$, where $s^A = \{a \mid a : s^{A'}\}$ and f^A is the restriction of $f^{A'}$ to ${s_1}^A \times \ldots \times {s_n}^A$; the above condition guarantees that the interpretation of the function symbols in A yields total functions. It is easy to check that A' satisfies $\mu_{Sen\,\Sigma}(\xi)$ iff A satisfies ξ.

Note that one-sorted algebras differing only on elements which do not satisfy any typing predicate represent the same many-sorted algebra.

Thus for every many-sorted signature Σ, a homogeneous signature $\mu_{Sign}(\Sigma)$ and two functions are defined: $\mu_{Sen\,\Sigma}$, which translates many sorted equational conditional sentences on Σ into homogeneous conditional sentences on $\mu_{Sign}(\Sigma)$ built on typing predicates and equalities, and $\mu_{Mod\,\Sigma}$, which partially translates homogeneous first-order structures on $\mu_{Sign}(\Sigma)$ into many-sorted algebras on Σ and is surjective, as it is immediate to check.

Since the change of notation, via signature morphisms, has a great relevance in the algebraic approach, being used for example to bind the actual to the formal parameters in parameterized specifications and to "put theories together to make specifications", the compatibility between the coding functions $\mu_{Sen\,\Sigma}$ and $\mu_{Mod\,\Sigma}$ defined for any signature Σ and the changes of notation has to be investigated.

Let $\bar{\sigma}\colon \Sigma_1 \to \Sigma_2$ be a morphism of many-sorted signatures. Then $\bar{\sigma}$ naturally induces a homogeneous signature morphism $\mu_{Sign}(\bar{\sigma}) = (\psi', \pi')$ from $\mu_{Sign}(\Sigma_1)$ into $\mu_{Sign}(\Sigma_2)$, defined by $\psi'(f) = \phi(f)$ for any $f \in F$ and $\pi'(\_ : s) = \_ : \sigma(s)$ for any $s \in S$. It is easy to check that the translation of sentences is compatible with signature morphisms, i.e. that $\mu_{Sen\,\Sigma_2}(\bar{\sigma}(\xi)) = \mu_{Sign}(\bar{\sigma})(\mu_{Sen\,\Sigma_1}(\xi))$.

Instead the partiality of the translation of algebras makes the compatibility between the algebra translations and signature morphisms delicate. Indeed it is intuitive to expect that the translation along a signature morphism of a one-sorted algebra simulating a many-sorted algebra simulates the translation of that many-sorted algebra; more formally, recalling that algebras are translated along signature morphisms in a countervariant direction into their reduct, if $A' \in dom(\mu)_{\Sigma_2}$, then $A'|_{\mu_{Sign}(\bar{\sigma})} \in dom(\mu)_{\Sigma_1}$ and $(\mu_{Mod\,\Sigma_2}(A'))_{|\bar{\sigma}} = \mu_{Mod\,\Sigma_1}(A'|_{\mu_{Sign}(\bar{\sigma})})$.

But the converse of the first implication does not hold, i.e. $A'|_{\mu_{Sign}(\bar{\sigma})} \in dom(\mu)_{\Sigma_1}$ does not imply $A' \in dom(\mu)_{\Sigma_2}$, as illustrated by the following example.

Let Σ_2 be the many-sorted signature

sig Σ_2 =
  sorts *nat*
  opns
    $0\colon \to nat$
    $inc, dec\colon nat \to nat$

Σ_1 be its subsignature where the *dec* operation has been dropped

sig Σ_1 =
  sorts *nat*
  opns
    $0\colon \to nat$
    $inc\colon nat \to nat$

and $\bar{\sigma}$ be the embedding of Σ_1 into Σ_2.

Consider now the one sorted algebra A' on $\mu_{Sign}(\Sigma_2)$, defined by

Algebra $A' =$

$$\begin{array}{l} |A'| = \mathbb{Z} \\ a : nat^{A'} \iff a \in \mathbb{N} \\ 0^{A'} = 0 \\ inc^{A'}(x) = x + 1 \\ dec^{A'}(x) = x - 1 \end{array}$$

Then $A' \notin dom(\mu)_{\Sigma_2}$, because $0 : nat$ holds, but $dec^{A'}(0) : nat$ does not and hence $dec^{A'}$ on appropriately typed input yields an untyped output. However $A'_{|\bar{\sigma}}$ is the same as A' but dec has been dropped, hence it obviously belongs to $dom(\mu)_{\Sigma_1}$.

Therefore a weaker condition (called *partial naturality*) has to be required for algebras than the one for sentences: if $A' \in dom(\mu)_{\Sigma_2}$, then $A'_{|\mu_{Sign}(\bar{\sigma})} \in dom(\mu)_{\Sigma_1}$ and $(\mu_{Mod_{\Sigma_2}}(A'))_{|\bar{\sigma}} = \mu_{Mod_{\Sigma_1}}(A'_{|\mu_{Sign}(\bar{\sigma})})$. $\square$

3.1.2 Simulations of Basic Specifications

Abstracting from the above construction, the general aspects of the coding of a *new* (many-sorted) into an *old* (one-sorted) formalism are:

- to each *new* signature an *old* signature corresponds;
- to each *new* sentence an *old* sentence corresponds;
- not any *old* algebra represents a *new* one, but to each *new* algebra at least one *old* corresponds, so that *old* algebras are (partially) translated by a surjective mapping.

This scheme generalizes to the frame of institutions by lifting maps to the proper categorical objects, taking care of the delicate points due to the partiality of model translation, and requiring that the only non-categorical structure, i.e. the validity relation, is preserved by them.

Since models are partially mapped, the usual notion of natural transformation is insufficient to describe the translation of the (*old*) model functor; thus "partially"-natural transformations are introduced to explicitly deal with the partiality of the model translation.

Def. 3.1.2 Let $\mathcal{I} = (\mathbf{Sign}, Sen, Mod, \models)$ and $\mathcal{I}' = (\mathbf{Sign}', Sen', Mod', \models')$ be institutions. Then a *simulation* $\mu\colon \mathcal{I} \to \mathcal{I}'$ consists of

- a functor $\mu_{Sign}: \mathbf{Sign} \to \mathbf{Sign}'$;
- a natural transformation $\mu_{Sen}: Sen \to Sen' \circ \mu_{Sign}$, i.e. a natural family of functions $\mu_{Sen_\Sigma}: Sen(\Sigma) \to Sen'(\mu_{Sign}(\Sigma))$, and
- a surjective *partially-natural* transformation

$$\mu_{Mod}: Mod' \circ \mu_{Sign} \to Mod,$$

  that is a family of functors $\mu_{Mod_\Sigma}: dom(\mu)_\Sigma \to Mod(\Sigma)$, where $dom(\mu)_\Sigma$ is a (non-necessarily full) subcategory of $Mod'(\mu_{Sign}(\Sigma))$ s.t.

  - μ_{Mod_Σ} is surjective on $|Mod(\Sigma)|$;
  - the family is partially-natural, i.e. for any signature morphism $\sigma \in \mathbf{Sign}(\Sigma_1, \Sigma_2)$

$$Mod(\sigma) \circ \mu_{Mod_{\Sigma_2}} = [\mu_{Mod_{\Sigma_1}} \circ Mod'(\mu_{Sign}(\sigma))]_{|dom(\mu)_{\Sigma_2}}$$

s.t. the following *satisfaction condition* holds:

$$A \models \mu_{Sen_\Sigma}(\xi) \iff \mu_{Mod_\Sigma}(A) \models \xi$$

for all $\Sigma \in |\mathbf{Sign}|$, all $A \in |dom(\mu)_\Sigma|$ and all $\xi \in Sen(\Sigma)$. □

Note that the partial-naturality condition implies

$$Mod'(\mu_{Sign}(\sigma))(dom(\mu)_{\Sigma_2}) \subseteq dom(\mu)_{\Sigma_1}.$$

In the sequel, for any simulation μ, the symbol μ will be used to denote its components, too, if the context makes clear the nature of the component.

Following the intuition that simulations corresponds to some sort of *implementation* or coding of a unknown formalism into a well known one, in the sequel the domain $\mathcal{I}$ of a simulation $\mu: \mathcal{I} \to \mathcal{I}'$ will be possibly referred to as *higher-level* or *new* and accordingly to $\mathcal{I}'$ as *lower-level* or *old.*

It is easy to check that $\mu: \mathcal{MS} \to \mathcal{L}$, whose components were informally sketched in Example 3.1.1, is a simulation, from now on denoted μ^M (the superscript M stands for *Many-sorted*) in order to reserve the symbol μ to denote a generic simulation.

It is possible to use simulations as morphisms in a category, by defining the composition of simulations componentwise and the identical simulation, whose components are identities. Hence, taking care of avoiding foundation problems, categories of institutions can be defined with simulations as morphisms, so that some usual concepts, like subobject, product and sum, are implicitly defined too.

Def. 3.1.3 Let $\mathcal{I} = (\mathbf{Sign}, Sen, Mod, \models)$, $\mathcal{I}' = (\mathbf{Sign}', Sen', Mod', \models')$ and $\mathcal{I}'' = (\mathbf{Sign}'', Sen'', Mod'', \models'')$ be institutions, $\mu: \mathcal{I} \to \mathcal{I}'$, where $\mu = (\mu_{Sign}, \mu_{Sen}, \mu_{Mod})$, and $\nu: \mathcal{I}' \to \mathcal{I}''$, where $\nu = (\nu_{Sign}, \nu_{Sen}, \nu_{Mod})$, be simulations.

Then $\nu \circ \mu: \mathcal{I} \to \mathcal{I}''$ is the composition componentwise, i.e. it consists of:

- $(\nu \circ \mu)_{Sign} = \nu_{Sign} \circ \mu_{Sign}$;
- $(\nu \circ \mu)_{Sen\,\Sigma} = \nu_{Sen\,\mu_{Sign}(\Sigma)} \circ \mu_{Sen\,\Sigma}$;
- $(\nu \circ \mu)_{Mod\,\Sigma} = \mu_{Mod\,\Sigma} \circ \nu_{Mod\,\mu_{Sign}(\Sigma)}$.

Moreover $\iota^{\mathcal{I}} = (\iota^{\mathcal{I}}{}_{Sign}, \iota^{\mathcal{I}}{}_{Sen}, \iota^{\mathcal{I}}{}_{Mod})$ is the simulation from $\mathcal{I}$ into $\mathcal{I}$ defined by:

- $\iota^{\mathcal{I}}{}_{Sign}$ is the identity functor on **Sign**;
- $\iota^{\mathcal{I}}{}_{Sen} = \{Id_{Sen(\Sigma)}\}_{\Sigma \in |\mathbf{Sign}|}$, where $Id_{Sen(\Sigma)}$ is the identity function on $Sen(\Sigma)$;
- $\iota^{\mathcal{I}}{}_{Mod} = \{Id_{Mod(\Sigma)}\}_{\Sigma \in |\mathbf{Sign}|}$, where $Id_{Mod(\Sigma)}$ is the identity functor on $Mod(\Sigma)$. □

It is obvious from the definition that the composition of simulation is a simulation too, that $\iota^{\mathcal{I}}$ is the identity simulation, i.e. that $\iota^{\mathcal{I}} \circ \mu = \mu = \mu \circ \iota^{\mathcal{I}'}$, and that simulations are associative, i.e. that $(\mu \circ \nu) \circ \xi = \mu \circ (\nu \circ \xi)$; thus it is possible to use simulations as morphisms in a category.

Some simple examples are sketched here in order to get a better intuitive understanding of the notion of simulation. More examples are disseminated in the next chapters.

Example 3.1.4 The equality between meta-terms can be interpreted by the identity, i.e. two terms are equal iff are denotations of the same concrete object, or more generally by a *congruence relation*, that is an equivalence relation preserving the structure of the working environment, for abstract data types usually the functional application. This second notion is used, for example in [35, 25], in order to represent the concept of *implementation* or *representation*; indeed in such a way more models are allowed, through an abstraction process, where different objects are regarded as equal on the base of some criteria.

The two notions are strictly related from an intuitive point of view and this relationship can be formalized by the definition of a simulation from the institution where equality is interpreted as identity into the institution where equality is represented by a predicate. In the sequel the simulation of (total) many-sorted algebras by typed logic is presented, but it is immediate to generalize this very construction to any "algebraic" case, where here "algebraic" is loosely used to denote the formalisms with an explicit notion of sort and operation (correspondingly of carrier and function) and sentences built on terms, including the total, partial, order-sorted algebras with or without predicates.

The basic idea is to introduce a binary predicate to represent the equality, substitute this symbol for the identity and translate back each algebra where the equality predicate is interpreted as a congruence into its quotient.

- let ${\mu^=}_{Sign}: \mathbf{Sign}_{\mathcal{MS}} \to \mathbf{Sign}_{\mathcal{TL}}$ be the functor associating each many-sorted signature (S, F) with the typed logic signature $(S, F, \{eq_s : ss\}_{s \in S})$ and each many-sorted signature morphism (σ, ϕ) with the typed logic signature morphism (σ, ϕ, π), defined by $\pi(eq_s) = eq_{\sigma(s)}$ for all $s \in S$;

- let ${\mu^=}_{Sen}: Sen_{\mathcal{MS}} \to Sen_{\mathcal{TL}}$ be the natural transformation translating each equality symbol between terms t and t' of sort s into $eq_s(t, t')$;

- let ${\mu^=}_{Mod}: Mod_{\mathcal{MS}} \to Mod_{\mathcal{TL}}$ be defined by:
  let $\Sigma = (S, F)$ be a many sorted signature;

  - the domain of ${\mu^=}_{Mod\Sigma}$ is the full subcategory of $Mod_{\mathcal{TL}}(\Sigma)$ whose objects A satisfy the following axioms:

$$\begin{aligned} eq_s(x, x) \\ eq_s(x, y) \supset eq_s(y, x) \\ eq_s(x, y) \wedge eq_s(y, z) \supset eq_s(x, z) \\ eq_{s_1}(x_1, y_1) \wedge \ldots \wedge eq_{s_n}(x_n, y_n) \supset \quad eq_s(f(x_1, \ldots, x_n), f(y_1, \ldots, y_n)) \end{aligned}$$

  - for any $A' \in |dom({\mu^=})_\Sigma|$ the translation $A = {\mu^=}_{Mod\Sigma}(A')$ consists of:
    - the carriers of A are the quotients of the carriers of A' w.r.t. the equality predicate: $s^A = s^{A'}/{eq_s}^{A'}$ for all $s \in S$;
    - the interpretation of function symbols goes through the classes: $f^A([a_1], \ldots, [a_n]) = [f^{A'}(a_1, \ldots, a_n)]$ for all $f \in F$;

    for any algebra morphism $h': A' \to B'$ the translation $h = {\mu^=}_{Mod\Sigma}(h')$ is defined on equivalence classes by $h([a]) = [h'(a)]$

It is easy to check that ${\mu^=}_{Sen}$ is natural, being a renaming, that ${\mu^=}_{Mod}$ is partially-natural, because its domains are model classes of theories and that ${\mu^=}_{Mod}$ is surjective, because each algebra A is represented by the first-order structure whose algebraic component is A and the eq_s predicates are interpreted by the identity; thus $\mu^=$ is a simulation.

Another example reminiscent of the concept of implementation is the simulation representing the implementation of a concrete data type (i.e. an algebra) by another one (possibly on a different signature).

Example 3.1.5 To sketch this simulation, the institution (without sentences) $\mathcal{I}(A, \Sigma)$ representing an algebra A on a signature Σ is introduced first.

Let A be a many-sorted algebra on a signature $\Sigma = (S, F)$; the institution $\mathcal{I}(A, \Sigma)$ consists of a category $\mathbf{Sign}_\Sigma$, that has strings of sorts as objects and k-tuple of terms t_i of sort s_i on variables $x_1, \ldots, x_n$ of sorts $s'_1, \ldots, s'_n$ respectively as arrows from $s'_1, \ldots, s'_n$ into $s_1, \ldots, s_k$, with substitution as composition, of the empty natural transformation of sentences (and empty satisfaction) and of the model functor $Mod_A \colon \mathbf{Sign}_\Sigma \to \mathbf{Cat}^{Op}$ yielding for each sort stream $s_1, \ldots, s_n$ the set ${s_1}^A \times \ldots \times {s_n}^A$ and for each k-tuple of terms the evaluation function.

The implementation of a many-sorted algebra A on a signature Σ by a many-sorted algebra A' on Σ' is formalized by a simulation from $\mathcal{I}(A, \Sigma)$ into $\mathcal{I}(A', \Sigma')$, where the signature component represents the abstraction function associating each "higher-level" syntactic object with a derived one at the "lower-level" and the model component represents the association of concrete objects with their abstract correspondent. The functoriality of the signature components guarantees that the functionality is preserved by the abstraction of the syntax; the naturality of the model component represents, in algebraic language, the intuition that the translation of elements is homomorphic w.r.t. the "higher-level" syntax, while surjectivness and partiality of simulations correspond to each "higher-level" element being represented by at least a "lower-level" one and non all "lower-level" elements being significant as "higher-level" objects.

3.2 Relationships between Institutions

In the literature it is often claimed that a frame is *equivalent* to another one, usually in the sense that both solve the same kind of problems, or that in both the results are equivalently (un)satisfactory. But the meaning of equivalence is usually not formally defined and quite often used to denote different levels of relationship. Indeed three different levels can be distinguished (and formalized by means of simulations), depending on whether the correspondence is between models, or categories of models, or specifications (theories).

At the *set-theoretic* level, for every model in the new frame a model in the old frame can be found that represents the given one, in the sense that it satisfies the same formulas, or, more precisely, that it satisfies corresponding formulas. This is formalized by requiring that there exists a simulation from the new into the old frame (s.t. the domains of the model component corresponds to any, possibly non-full, subcategory of the old models). At this level most properties are missing, in particular no structured way of defining models is guaranteed to be preserved, because it usually involves categorical constructions. To have a *categorical* correspondence between two frames, at least the domain of the simulation has to be a full subcategory of the old models; moreover some more properties have to be required depending on the categorical structures that are intended to be preserved. Here the focus is on the initial structures and minimal conditions are given to preserve initiality. Even if there is a categorical simulation, the power of the specification languages in the two frames can be quite different; in particular it is possible that in the new frame some categories are definable by sets of sentences that are not so in the old one (and vice versa). To guarantee that the relationship is at the *logical level*, i.e. for every specification (i.e. the class of models which satisfy a set of sentences) in the new frame there exists a specification in the old frame equivalent to the given one in the categorical sense, it has to be required not only that the domain of the model component is a full subcategory of the category of old models, but also that it is described by a set of old sentences.

3.2.1 A Paradigmatic Example: Partial versus Total Specifications

Here the use of the notion of simulation, with its various specializations, is illustrated as a tool for understanding the relationship between two formalisms with respect to the solution of a problem. As a paradigmatic example, the specification of (strict) partial functions in a partial and a total frame has been chosen, picking up from the discussion in Chapter 2 just few representative cases.

The following analysis, though not pretending to be exhaustive especially on the pragmatic side, will highlight the subtleties of the relationship between the two frames and possibly reveal some misbeliefs.

Semantic Level

The relationship between partial and total frames is first analyzed from a semantic point of view, i.e. disregarding their logics. Formally this means that the institutions are *without sentences*.

Def. 3.2.1 Let $\mathcal{PAR}_0$ denote the institution $(\mathbf{Sign}_{\mathcal{PAR}}, \emptyset, Mod_{\mathcal{PAR}}, \emptyset)$ of partial algebras without sentences and $\mathcal{MS}_0$ denote the institution

$(\mathbf{Sign}_{\mathcal{MS}}, \emptyset, Mod_{\mathcal{MS}}, \emptyset)$ of total many-sorted algebras without sentences. $\square$

In the algebraic community there is a widespread belief that partiality can also be handled without explicit partial functions, in the usual total frame, simply by introducing a distinguished constant $\perp$ (one for each sort) to represent the undefined computations; in this way to any partial algebra A its trivial totalization corresponds (see e.g. the *error algebras* for a more sophisticated version of this idea). Following this intuition it is possible to define a *simulation* $\mu_0^\perp$ of partial by total algebras, where every partial algebra is simulated by its trivial totalization; but some homomorphisms between the trivial totalizations of partial algebras cannot be translated into homomorphisms of partial algebras, because the image of some *defined* element (i.e. of elements different from $\perp$) may be *undefined* (i.e. equal to $\perp$), while the homomorphisms of partial algebras are *total* functions. Therefore the domain of the simulation is not a full subcategory of the models and hence most categorical properties are missing.

Def. 3.2.2 The simulation $\mu_0^\perp: \mathcal{PAR}_0 \to \mathcal{MS}_0$ consists of:

- $\mu_0^\perp{}_{Sign}: \mathbf{Sign}_{\mathcal{PAR}} \to \mathbf{Sign}_{\mathcal{MS}}$ is defined by $\mu_0^\perp{}_{Sign}(S, F) = (S, F')$, where if $w \neq \emptyset$, then $F'_{w,s} = F_{w,s}$ else $F'_{\emptyset,s} = F_{\emptyset,s} \cup \{\perp_s\}$, and by $\mu_0^\perp{}_{Sign}(\sigma, \phi) = (\sigma, \phi')$,where $\phi'(f) = \phi(f)$ for any $f \in F_{w,s}$ and $\phi(\perp_s) = \perp_{\sigma(s)}$.

- $\mu_0^\perp{}_{Sen}: \emptyset \to \emptyset$ is the empty natural transformation;

- $\mu_0^\perp{}_{Mod}: Mod_{\mathcal{MS}} \circ \mu_0^\perp{}_{Sign} \to Mod_{\mathcal{PAR}}$ is defined by:

  - $dom(\mu_0^\perp)_\Sigma$ is the subcategory of $Mod_{\mathcal{MS}}(\mu_0^\perp{}_{Sign}(\Sigma))$ whose objects are the total algebras A' s.t. $f^{A'}(a_1, \ldots, a_n) \neq \perp_s$ implies $a_i \neq \perp^{A'_{s_i}}$ for all $i = 1 \ldots n$ for every $f \in F_{s_1 \ldots s_n, s}$ (*strictness*) and whose arrows are the many-sorted homomorphisms $h' \in Mod_{\mathcal{MS}}(\mu_0^\perp{}_{Sign}(\Sigma))(A', B')$ s.t. $a \neq \perp_s^{A'}$ implies $h'_s(a) \neq \perp_s^{A'}$ for any $s \in S$.
  - for each $A' \in dom(\mu_0^\perp)_\Sigma$ the partial algebra $A = \mu_0^\perp{}_{Mod\Sigma}(A')$ consists of $s^A = s^{A'} - \{\perp_s^{A'}\}$ for every $s \in S$ and for every $f \in F_{s_1 \ldots s_n, s}$ and every $a_i \in s_i^A$ for $i = 1 \ldots n$, if $f^{A'}(a_1, \ldots, a_n) \neq \perp_s^{A'}$, then $f^A(a_1, \ldots, a_n) = f^{A'}(a_1, \ldots, a_n)$, else $f^{A'}(a_1, \ldots, a_n)$ is undefined;
  - $\mu_0^\perp{}_{Mod\Sigma}(h')$ is the restriction of h' to $s^{\mu_0^\perp{}_{Mod\Sigma}(A')}$. $\square$

Note that formally strictness is not needed, because there are no sentences whose validity has to be preserved; however it is preferable to require the strictness condition, because it is more intuitive specifying strict partial algebras and it will be needed in the sequel, to deal with logics.

Although obviously A' and $\mu_0^{\perp}{}_{Mod\Sigma}(A')$ are strictly related from a set theoretic point of view, the correspondence, due to the domain of $\mu_0^{\perp}$ being a non-full subcategory, is not adequate for categorical purposes, in particular the initial model is not preserved by $\mu_0^{\perp}$.

Indeed in both frames the initial model is characterized by the *no junk* and *no confusion* conditions of [64], which mean that every element is denoted by some term and that two ground terms are equal in the initial object iff they are equal in every algebra of the class (in the partial frame the existential equality is considered, holding if both sides denote the same element of the carrier, so that also the *minimal definedness* holds). Thus the minimal equality (no-confusion) of the initial model in the total frame implies, in particular, the minimal equality with $\perp$ and hence the *maximal* definedness of its translation; therefore in most cases the translation of the initial model is not initial.

Categorical Level

Another way of coding partiality in terms of total algebras is to split every carrier by a typing predicate in typed (i.e. defined) and untyped elements and to represent every partial function by a total one which results in an untyped element over every input outside its domain (for similar approaches see e.g. [60], where one-sorted total algebras are used, [69] and [70]). Moreover, in order to handle logical formulas in the sequel, a binary predicate, that plays the role of the existential equality, and holds on a' and b' iff a' and b' are equal and appropriately typed is introduced. The corresponding simulation is as follows.

Def. 3.2.3 Let $\mathcal{TL}_0$ denote the institution $(\mathbf{Sign}_{\mathcal{TL}}, \emptyset, Mod_{\mathcal{TL}}, \emptyset)$ of typed first-order structures (total many-sorted algebras with predicates) without sentences.

The simulation $\mu^P{}_0: \mathcal{PAR}_0 \to \mathcal{TL}_0$ consists of:

- $\mu^P{}_{0Sign}: \mathbf{Sign}_{\mathcal{PAR}} \to \mathbf{Sign}_{\mathcal{TL}}$ consists of $\mu^P{}_{0Sign}(S, F) = (S', F', P')$, where $S' = S$, $F' = F$ and if $w = ss$, then $P'_{ss} = \{eq_s\}$, if $w = s$, then $P'_s = \{D_s\}$, otherwise $P'_w = \emptyset$, and $\mu^P{}_{0Sign}(\sigma, \phi) = (\sigma', \phi', \pi')$,where $\sigma' = \sigma$, $\phi' = \phi$, $\pi'(D_s) = D_{\sigma(s)}$ and $\pi'(eq_s) = eq_{\sigma(s)}$.
- $\mu^P{}_{0Sen}$ is the empty natural transformation;
- $\mu^P{}_{0Mod}: Mod_{\mathcal{TL}} \circ \mu^P{}_{0Sign} \to Mod_{\mathcal{PAR}}$ is defined by:
  - $dom(\mu^P{}_0)_\Sigma$ is the full subcategory of $Mod_{\mathcal{TL}}(\mu^P{}_{Sign}(\Sigma))$ whose objects are the total algebras A' s.t. for every $f \in F_{s_1 \ldots s_n, s}$ if ${D_s}^{A'}(f^{A'}(a_1, \ldots, a_n))$ holds, then ${D_{s_i}}^{A'}(a_i)$ holds, too, for every $i = 1 \ldots n$ (*strictness*) and $eq_s^{A'}(a, b)$ iff $a = b$ and ${D_s}^{A'}(a)$, ${D_s}^{A'}(b)$.

- for every $A' \in dom(\mu^P)_\Sigma$ the partial algebra $A = {\mu^P}_{0Mod_\Sigma}(A')$ consists of $s^A = {D_s}^{A'}$ for every $s \in S$ and for every $f \in F$ and every $a_i \in {s_i}^A$ for $i = 1 \dots n$ if ${D_s}^{A'}(f^{A'}(a_1, \dots, a_n))$ holds, then $f^A(a_1, \dots, a_n) = f^{A'}(a_1, \dots, a_n)$, else $f^A(a_1, \dots, a_n)$ is undefined.
- for any $h \in dom({\mu^P}_0)_\Sigma(A', B')$ the arrow ${\mu^P}_{Mod_\Sigma}(h')$ is $h'_{|{\mu^P}_{Mod_\Sigma}(A')}$. □

Now initial models are translated along ${\mu^P}_0$ to initial models; the proof follows a pattern common to most algebraic frames. First it is shown that the translation I of an initial object is *weakly* initial (i.e. that there exists at least one arrow from I into any object), so that the *no-confusion* condition holds; the weak initiality comes from $dom({\mu^P}_0)_\Sigma$ being a full subcategory of the total models and ${\mu^P}_{0Mod}$ being surjective on the objects. Then I is shown to be term-generated, so that the *no-junk* condition holds, too, because the total initial object is term-generated and term-generatedness is preserved by ${\mu^P}_0$.

Abstracting from the two main points of the above proof technique, the *categorical* simulations are defined, which preserve "term-generatedness" and whose domains are full subcategories, and show that categorical simulations preserve initiality.

Def. 3.2.4 Let $\mathbf{C}$ be a category and c be an object of $\mathbf{C}$; then c is called *inductive* iff $\mathbf{C}(c, c')$ has at most one element for every $c' \in \mathbf{C}$. For every subcategory $\mathbf{C}'$ of $\mathbf{C}$, c is called *weakly initial* in $\mathbf{C}'$ iff $\mathbf{C}'(c, c')$ has at least one element for every $c' \in \mathbf{C}'$.

Let $\mathcal{I}$ and $\mathcal{I}'$ be institutions and μ be a simulation from $\mathcal{I}$ into $\mathcal{I}'$. Then μ is called *categorical* iff every $dom(\mu)_\Sigma$ is a full sub-category of $Mod'(\Sigma)$ and μ_{Mod_Σ} preserves the inductive objects of $Mod'(\mu_{Sign}(\Sigma))$ belonging to $dom(\mu)_\Sigma$. □

Note that the property of being categorical only involves the model components of simulations (and, implicitly, the translation of signatures); thus if two simulations coincide on signatures and models and the first is categorical, then also the second one is so, independently of the formulas that are chosen as sentences of the institutions and their translation.

In most algebraic frames the interesting classes of models are closed w.r.t. subalgebras and this guarantees that their initial models, if any, are term-generated; this can be generalized to every categorical frame, noting that the notion of subalgebra generalizes to the categorical concept of *regular subobject*.

Def. 3.2.5 Let $\mathbf{C}$ be a category and $f, g \in \mathbf{C}(A, B)$ be a pair of parallel arrows. Then an arrow $e \in \mathbf{C}(E, A)$ is an *equalizer* of f and g iff it satisfies the following conditions

- $f \circ e = g \circ e$ (*e equalizes f* and g);
- for any $k \in \mathbf{C}(K, A)$ s.t. $f \circ k = g \circ k$ there exists a unique $\pi \in \mathbf{C}(K, E)$ s.t. $e \circ \pi = k$ (*k factorizes through e*).

If $e \in \mathbf{C}(E, A)$ is an equalizer of some f and g, then E is a *regular subobject* of A. □

Lemma 3.2.6 Let $\mathbf{C}$ be a category having equalizers and $\mathbf{C}'$ be a subcategory of $\mathbf{C}$ closed under equalizers and regular subobjects. Then I is initial in $\mathbf{C}'$ only if I is inductive in $\mathbf{C}$.

Proof. Let I be initial in $\mathbf{C}'$, assume that there exist $f, g \in \mathbf{C}(I, A)$ for some $A \in |\mathbf{C}|$ and show that $f = g$. Since $\mathbf{C}$ has equalizers, there exists the equalizer $e \in \mathbf{C}(E, I)$ of f and g. Since $\mathbf{C}'$ is closed w.r.t. regular subobjects, $E \in |\mathbf{C}'|$ and hence there exists a unique $!_E \in \mathbf{C}(I, E)$ and $e \circ !_E = Id_I$, because the identity is the unique arrows from I into I, as I is initial. Thus $f = f \circ (e \circ !_E) = (f \circ e) \circ !_E$ and analogously $g = (g \circ e) \circ !_E$; since e is the equalizer of f and g, $f \circ e = g \circ e$ and hence $f = (f \circ e) \circ !_E = (g \circ e) \circ !_E = g$, i.e. $f = g$. □

It is worth to note that both the institutions of partial as well as total many-sorted algebras, with or without predicates, do have equalizers, and that the model classes of positive Horn-clauses are closed w.r.t. regular subobjects (in general they are not closed w.r.t. generic subobjects); thus the following proposition applies in most cases.

Prop. 3.2.7 Let $\mathcal{I} = (\mathbf{Sign}, Sen, Mod, \models)$, $\mathcal{I}' = (\mathbf{Sign}', Sen', Mod', \models')$ be institutions s.t. for any $\Sigma' \in |\mathbf{Sign}'|$ the category $Mod'(\Sigma')$ has equalizers and μ be a categorical simulation from $\mathcal{I}$ into $\mathcal{I}'$. If I' is initial in a full subcategory $\mathbf{C}'$ of $dom(\mu)_\Sigma$ closed w.r.t. regular subobjects (performed in $Mod'(\mu_{Sign}(\Sigma))$), then $\mu_{Mod\Sigma}(I')$ is initial in $\mu_{Mod\Sigma}(\mathbf{C}')$.

Proof. To show that $I = \mu_{Mod\Sigma}(I')$ is initial in $\mathbf{C} = \mu_{Mod\Sigma}(\mathbf{C}')$, it is sufficient to prove that I is weakly initial in $\mathbf{C}$ and that it is inductive, so that $\mathbf{C}(I, A)$ has exactly one element. Since $dom(\mu)_\Sigma$ is a full subcategory of $Mod'(\mu_{Sign}(\Sigma))$, and I' is initial in $\mathbf{C}'$ for any $A = \mu_{Mod\Sigma}(A') \in |\mathbf{C}|$ there exists $\mu_{Mod\Sigma}(!_{A'}) \in \mathbf{C}(I, A)$, where $!_{A'}$ is the unique arrows from the initial object I' into A', and hence I is weakly initial in $\mathbf{C}$. Because of the above Lemma 3.2.6, I' is inductive and hence I is inductive, too, because categorical simulations preserve inductive objects. □

Cor. 3.2.8 Let $\mathbf{C}'$ be a full subcategory of $dom({\mu^P}_0)_\Sigma$ closed w.r.t. subalgebras and I' be the initial object in $\mathbf{C}'$. Then ${\mu^P}_{0Mod}(I')$ is initial in ${\mu^P}_{0Mod}(\mathbf{C}')$.

Proof. Since inductive in $Mod_{\mathcal{MS}}({\mu^P}_{0Sign}(\Sigma))$ coincides with term-generated and ${\mu^P}_0$ does not introduce function symbols, ${\mu^P}_0$ preserves inductive objects; moreover its domain is a full subcategory by definition. Therefore ${\mu^P}_0$ is categorical; moreover $Mod_{\mathcal{MS}}({\mu^P}_{0Sign}(\Sigma))$ has equalizers, which are (usual) subalgebras, and hence the Prop. 3.2.7 applies. $\square$

Logical Level.

Consider now the logical aspect of partial and total frames and investigate the equivalences of their expressive power. In the total [partial] frame the institution $\mathcal{TL}$ [$\mathcal{PPAR}$] of positive Horn-clauses [built on existential equality] and its subinstitution $\mathcal{GTL}$ [$\mathcal{GPAR}$], where the sentences are without variables, are considered.

Consider first the trivial simulation $\mu_0^\perp$. Let A' be in $dom(\mu_0^\perp)_\Sigma$ and consider a ground existential equality $t = t'$; then $A = \mu_0^\perp{}_{Mod\Sigma}(A')$ satisfies $t = t'$ iff t^A and t'^A denote the same element of $s^A = s^{A'} - \{\perp_s\}$, i.e. iff $t^{A'} = t'^{A'} \neq \perp_s$; thus to generalize $\mu_0^\perp$ to a simulation from $\mathcal{GPAR}$, a stronger (and unusual) logic than the positive Horn-clauses is needed in the total frame. Therefore the trivial totalization fails in both the categorical and the logical aspects, in the sense that, although it is true that any partial algebra is equivalent from a set theoretic point of view to its trivial totalization, the equivalence becomes false if algebra morphisms are considered; moreover it relates Horn-Clauses to a more powerful first-order fragment.

Consider now the simulation ${\mu^P}_0$. Every ground Horn-clause is naturally translated into the total frame, just by replacing every existential equality symbol with the corresponding predicate eq. However if variables appear in the formula, this translation from the partial to the total frame does not preserve the validity of sentences. Indeed, consider for example $D_s(x)$; then obviously any partial algebra satisfies it (undefined elements do not exist), while some total algebras in the domain of ${\mu^P}_0$ do not, because valuations of variables in the total frame range also over the elements which do not satisfy the definedness predicates (and hence are dropped by the simulation). More generally the valuations for the total frame which range over undefined elements must not be taken in account, in order to establish the validity of translations of partial sentences. To overcome this problem it is sufficient to add to the premises of every sentence the definedness assertions for each of its variables, so that every valuation s.t. $V(x) = a$ and $\neg D(a)$ satisfies the sentence, because one of the premises is false; thus the validity only depends on "defined" valuations also in the total case. Note that in this way equations with variables in the partial frame are translated into conditional axioms of the total formalism. This, together with the fact that the simulation ${\mu^P}_0$ (properly generalized to deal with sentences) is categorical, illustrates the deep reason for the model classes of partial equational specifications being quasi-varieties (see [92]), like the model classes of total conditional specifications, and not varieties, as the model classes of total equational specifications are (see [64]).

Def. 3.2.9 The categorical simulation $\mu^P : \mathcal{PPAR} \to \mathcal{TL}$ coincides with ${\mu^P}_0$ on signatures and models, and on sentences is defined by

$$\mu^P{}_{Sen\,\Sigma}(\xi) = D_{s_1}(x_1) \wedge \ldots \wedge D_{s_k}(x_k) \wedge eq_{s'_1}(t_1, t'_1) \wedge \ldots \wedge eq_{s'_n}(t_n, t'_n) \supset eq_s(t, t')$$

where $\xi = (t_1 = t'_1 \wedge \ldots \wedge t_n = t'_n \supset t = t')$ and $x_1, \ldots, x_k$ are the variables of ξ. □

Since the domain of μ^P is the model class of the following axioms $th(\mu^P)$:

$$D_s(f(x_1, \ldots, x_n)) \supset D_{s_i}(x_i) \text{ for } i = 1 \ldots n \ (strictness) \text{ and}$$

$$D_s(x) \wedge D_s(y) \wedge x = y \Leftrightarrow eq_s(x, y), \text{ i.e.}$$

$D_s(x) \wedge D_s(y) \wedge x = y \supset eq_s(x, y)$, $eq_s(x, y) \supset D_s(x)$, $eq_s(x, y) \supset D_s(y)$ and $eq_s(x, y) \supset x = y$, every model class of a partial *presentation* (Σ, Ax) is simulated by the model class of the total presentation

$$(\mu^P{}_{Sign}(\Sigma), \mu^P{}_{Sen\,\Sigma}(Ax) \cup th(\mu^P)).$$

This kind of simulation, translating presentations into presentations is called *logical*.

Def. 3.2.10 Let $\mathcal{I} = (\mathbf{Sign}, Sen, Mod, \models)$, $\mathcal{I}' = (\mathbf{Sign}', Sen', Mod', \models')$ be institutions and μ be a simulation from $\mathcal{I}$ into $\mathcal{I}'$. Then μ is called *logical* iff $dom(\mu)_\Sigma$ is the full subcategory of $Mod'(\mu_{Sign}(\Sigma))$ whose objects are the model class of a set $th(\mu)_\Sigma \subseteq Sen'(\mu_{Sign}(\Sigma))$ of sentences for every $\Sigma \in |\mathbf{Sign}|$. □

Although in general the family $\{th(\mu)_\Sigma\}_{\Sigma \in |\mathbf{Sign}|}$ is not functorial, i.e. it is not possible to define a sub-functor F of Sen s.t. $F(\Sigma) = th(\mu)_\Sigma$, if the family of their closures under logical consequences $\{th(\mu)^\bullet_\Sigma\}_{\Sigma \in |\mathbf{Sign}|}$ is considered, where $th(\mu)^\bullet_\Sigma = \{\alpha \mid A' \models' \alpha \text{ for all } A' \in dom(\mu)_\Sigma\}$, then the partial-naturality condition on $\{dom(\mu)_\Sigma\}_{\Sigma \in |\mathbf{Sign}|}$ guarantees the functoriality of $\{th(\mu)^\bullet_\Sigma\}_{\Sigma \in |\mathbf{Sign}|}$ and hence every logical simulation is a map of institutions, too (see [63]).

Since the translations via μ^P of ground partial Horn Clauses are ground Horn Clauses, μ^P can be specialized to a simulation between the institutions $\mathcal{GPAR}$ and $\mathcal{GTL}$. It is still categorical, but is not logical anymore; indeed axioms with variables are needed to define the domain, as the following example shows.

Example 3.2.11 For any signature $\Sigma = (S, F) \in |\mathbf{Sign}_{\mathcal{PAR}}|$ with at least one function symbol there does not exist a set $th' \subseteq Sen'(\mu^P{}_{Sign}(\Sigma))$ of ground sentences s.t. $dom(\mu^P)_\Sigma$ is the class of models of th'. Indeed an algebra A' exists belonging to the model class of any set of ground Horn-clauses that is not in $dom(\mu^P)_\Sigma$. Let A' be defined by $s^{A'} = \{1_s, 2_s\}$ for all $s \in S$, $f^{A'}(x_1, \ldots, x_n) = 1_s$ for all $f \in F_{s_1 \ldots s_n, s}$, $eq_s^{A'} = \{(1_s, 1_s)\}$ and $D_s{}^{A'} = \{1_s\}$. Then for any ground term $t \in T_{\Sigma s}$, its evaluation in A' is 1_s and hence A' satisfies any ground formula. But A' does not belong to $dom(\mu^P)_\Sigma$, because functions are not strict; indeed $D_s{}^{A'}(f^{A'}(2_{s1}, \ldots, 2_{sn}))$ but $\neg D_{s_i}{}^{A'}(2_{s_i})$. □

3.2.2 Non-strict Specifications

The same technique of the last section can be applied also in the case of specification of non-strict functions, i.e. of functions that can yield a result also on incomplete inputs, like the ubiquitous *if_then_else*.

In the next subsections the results about non-strict don't care algebras are summarized and then in the following section the relationship between non-strict and total algebras is analyzed, showing that the the same pattern as for the partial adversus total case applies in this case, too.

Non-Strict Don't Care Algebras

In this approach to the algebraic specification of non-strict functions, (see e.g. [4, 5]), the basic idea is the one of *partial product*.
Usually the product $A_1 \times \ldots \times A_n$ is the set of all (total) functions g from $\{1 \ldots n\}$ into $A_1 \cup \ldots \cup A_n$ s.t. $g(i) \in A_i$. This concept is generalized by allowing *partial functions*.

In order to keep the notation as similar as possible to the usual one, the symbol ? is used to denote the "undefined" elements.

Def. 3.2.12 Let $A_1, \ldots, A_n$ be sets. The *partial product* $\times_p\{A_1, \ldots, A_n\}$ of $A_1, \ldots, A_n$ consists of all partial functions from $\{1, \ldots, n\}$ into $A_1 \cup \ldots \cup A_n$ s.t. if $g(i)$ is defined, then $g(i) \in A_i$. If $n \geq 2$, instead of $\times_p\{A_1, \ldots, A_n\}$ the infix notation $A_1 \times_p \ldots \times_p A_n$, reminiscent of the standard notation, is also used.

Over $\times_p\{A_1, \ldots, A_n\}$ is naturally defined a partial order $\leq$ by $a \leq b$ iff $a(i) \in A_i$ implies $b(i) =_e a(i)$ for all $i = 1 \ldots n$.

A partial function g from $\times_p\{A_1, \ldots, A_n\}$ into a set A is called *strict* iff $g(a) \in A$ implies $a(i) \in A_i$ for all $i = 1, \ldots, n$ and is called *monotonic* if $a \leq b$ and $g(a) \in A$ implies $g(a) =_e g(b)$.

In the sequel an element $a \in \times_p\{A_1, \dots, A_n\}$ is denoted by $(a_1, \dots, a_n)$, where $a_i = a(i)$ if $a(i) \in A_i$ and $a_i =?$ otherwise. □

Two remarks are in order here. First note that A and $\times_p\{A\}$ are not in general isomorphic; for example, if A has finite cardinality k, then in $\times_p\{A\}$ there is one more element, the totally undefined tuple, so that $\times_p\{A\}$ has cardinality $k+1$.

Moreover, while the usual product coincides with the categorical product in the category of sets with total functions as arrows, the *partial* product is not the categorical product in the category of sets with *partial* functions as arrows, because the uniqueness of the factorization through the partial product fails. Indeed consider a singleton set X and its binary *partial* product $Y = X \times_p X$, with projections $\pi_i(x) = x(i)$; the factorization through Y of the couple of functions $h = <\perp, \perp>$, where $\perp$ is the totally undefined function on X, is not unique, because $\pi_i \circ f = \perp = \pi_i \circ g$ for both $f, g\colon X \to Y$, respectively defined by $f(\cdot)$ is undefined and $g(\cdot)$ is the partial function x, where both $x(1)$ and $x(2)$ are undefined.

Def. 3.2.13 Let $\Sigma = (S, F)$ be a many-sorted signature; a *non-strict* Σ*-algebra* consists of a family $\{s^A\}_{s \in S}$ of sets, the *carriers*, and of a family $\{f^A\}_{f \in F_{w,s}, w \in S^*, s \in S}$ of partial functions, the *interpretations of operation symbols*, s.t. if $f \in F_{\Lambda,s}$, then either f^A is undefined or $f^A \in s^A$, otherwise $f \in F_{s_1 \dots s_n, s}$ with $n \geq 1$ and $f^A\colon {s_1}^A \times_p \dots \times_p {s_n}^A \to s^A$ is a monotonic function.

An algebra A is called *strict* if f^A is strict for any $f \in F$; moreover a strict algebra is called *total* if $\underline{a}(i) \in s_i^A$ for all $i = 1 \dots n$ implies $f^A(\underline{a}) \in s^A$ for all $f \in F_{s_1 \dots s_n, s}$. The class of all non-strict Σ-algebras will be denoted by $\mathrm{NSAlg}(\Sigma)$. □

Then, by definition, *strict algebras* are exactly the partial algebras and *total algebras* are the usual ones.

Note that in non-strict algebras no extra-elements are in the carriers to define non-strict functions. This on one hand corresponds to the intuition that functions like *if_then_else* are able to compute the result on incomplete input, but on the other side does not support the idea of (differentiating) error recovery or exception handling.

Depending on the partiality of the functions, there are several possibilities to define homomorphisms, each one being useful for a different purpose (see e.g. [22, 26, 80]). Here the choice follows the tradition of partial algebras (see e.g. [1, 26, 22, 92]), where they are used in order to get a no-junk&no-confusion initial object (see [64]).

Def. 3.2.14 Let $\Sigma = (S, F)$ be a signature, A and B be non-strict algebras over Σ. Then a *homomorphism* $h: A \to B$ is a family $\{h_s: s^A \to s^B\}_{s \in S}$ of total functions s.t. $f^A(\underline{a}) \in s^A$

implies $h_s(f^A(\underline{a})) = f^B(h \circ \underline{a})$, where $h \circ \underline{a}$ is defined by $h \circ \underline{a}(i) = h_{s_i}(\underline{a}(i))$ for $i = 1 \ldots n$, for all $f \in F_{s_1 \ldots s_n, s}$ and all $\underline{a} \in {s_1}^A \times_p \ldots \times_p {s_n}^A$.

The category $\mathbf{NSAlg}(\Sigma)$ is defined by:

- the objects of $\mathbf{NSAlg}(\Sigma)$ are $\mathrm{NSAlg}(\Sigma)$;
- the arrows in $\mathbf{NSAlg}(\Sigma)$ are all the homomorphisms;
- composition is done componentwise;
- the identity on A is $\{Id_{s^A}\}_{s \in S}$. $\square$

Note that each homomorphism between strict algebras is a total homomorphism of partial algebras and each homomorphism between total algebras is a usual total homomorphism; thus the category both of total algebras and of partial algebras with total homomorphisms are full sub-categories of $\mathbf{NSAlg}(\Sigma)$.

The term evaluation is defined like in the partial strict frame, but also partial valuations for variables have to be allowed. Valuations being partial functions, it is possible to define in a canonical way an order on them, with the empty valuation as minimal element.

Def. 3.2.15 Let $\Sigma = (S, F)$ be a signature, $X = \{X_s\}_{s \in S}$ be a family of S-sorted variables. For all algebras $A \in \mathrm{NSAlg}(\Sigma)$ and all *valuations* $V = \{V: X_s \to s^A\}_{s \in S}$ for X in A, where V is a partial function, the *evaluation* $eval^{A,V}: T_\Sigma(X) \to A$ is inductively defined by:

- $eval^{A,V}(x) = V(x)$ for all $x \in X$;
- $eval^{A,V}(f) = f^A$ for all $f \in F_{\Lambda,s}$;
- $eval^{A,V}(f(t_1, \ldots, t_n)) = f^A(\underline{a})$, where $\underline{a}(i) = eval^{A,V}(t_i)$ with $i = 1 \ldots n$ for all $f \in F_{s_1 \ldots s_n, s}$ and $t_i \in T_\Sigma(X)_{s_i}$ for $i = 1 \ldots n$.

Let $V, V': X \to A$ be valuations; then $V \leq V'$ iff $V(x) \in s^A$ implies $V'(x) =_e V(x)$ for all $x \in X$. The valuation $V_?$ for X in A is the empty map, i.e. $V_?(x)$ is undefined for all $x \in X_s$ and all $s \in S$.

In the sequel $eval^{A,V}(t)$ will be denoted by $t^{A,V}$; moreover if X is the empty set (so that there exists a unique valuation $V_?$ for X in A), $eval^{A,V}$ will be denoted simply by $eval^A$ and $eval^{A,V}(t)$ by t^A. Finally $\underline{a}$, defined by $\underline{a}(i) = {t_i}^{A,V}$ for $i = 1, \ldots n$, will be denoted by $({t_1}^{A,V} \ldots, {t_n}^{A,V})$. $\square$

It is worth to note that the order on the valuations is preserved by the evaluation, i.e. $V \leq V'$ implies $eval^{A,V} \leq eval^{A,V'}$.

Prop. 3.2.16 Let $\Sigma = (S, F)$ be a signature, A be a non-strict algebra over Σ, X be an S-sorted family of variables, V and V' be valuations for X in A s.t. $V \leq V'$. For all terms $t \in T_\Sigma(X)_s$ if $t^{A,V} \in s^A$ then $t^{A,V'} {=_e} t^{A,V}$.
Proof. By induction over the definition of terms, because operation symbols are interpreted by monotonic functions. □

In the total frame, the term-algebras are the free objects in the class of all total algebras, because of the uniqueness of the evaluation w.r.t. a valuation. Here, as in the partial case, term-algebras are not free, because the evaluations are not homomorphisms, being partial functions. However a derived property holds also in this case, although a bit relaxed. Indeed in the total frame the freeness of the term algebra implies that if the upper triangle of the diagram 1 commutes, then the triangle below commutes too. Analogously in this frame if $h \circ V \leq V'$, then $h \circ eval^{A,V} \leq eval^{A',V'}$, so that the diagram 2 is obtained.

$$\begin{tikzcd} & X \arrow[dl, "V"'] \arrow[dr, "V'"] & \\ A \arrow[rr, "h"] & & A' \\ & T_\Sigma(X) \arrow[ul, "eval^{A,V}"] \arrow[ur, "eval^{A',V'}"'] & \end{tikzcd}$$

diagram 1

$$\begin{tikzcd} & X \arrow[dl, "V"'] \arrow[dr, "V'"] & \\ A \arrow[rr, "h"] & & \leq A' \\ & T_\Sigma(X) \arrow[ul, "eval^{A,V}"] \arrow[ur, "eval^{A',V'}"'] & \end{tikzcd}$$

diagram 2

This result also holds in the partial frame and is crucial in order to get that the initial object in a class, if any, satisfies the *no-junk* and *no-confusion* properties (see [64]). Indeed the diagram 2 where A is the initial object and h the unique homomorphism from the initial object into A', states that each term defined in A has to be defined in A' (no-junk) and that two terms existentially equal in A have to be existentially equal in A' (no-confusion).

Prop. 3.2.17 Let $\Sigma = (S, F)$ be a signature and $X = \{X_s\}_{s \in S}$ be a family of S-sorted variables. For all non-strict algebras $A, A' \in \mathrm{NSAlg}(\Sigma)$, all valuations V for X in A and V' for X in A' and all homomorphisms $h\colon A \to A'$ s.t. $h \circ V \leq V'$

1. $t^{A,V} \in s^A$ implies $h(t^{A,V}) =_e t^{A',V'}$ for all $t \in T_\Sigma(X)$;

2. $t^{A,V} =_e t'^{A,V}$ implies $t^{A',V'} =_e t'^{A',V'}$ for all $t, t' \in T_\Sigma(X)$.

Proof. By induction over the definition of terms. □

An *ad hoc* definition of congruence and of quotient on term algebras is introduced, whose use is limited to the study of the existence of the initial model. From now on let X denote a family $X = \{X_s\}_{s\in S}$ of variables s.t. X_s is non-empty for all $s \in S$.

Def. 3.2.18 Let $\Sigma = (S, F)$ be a signature; a *congruence* $\equiv$ is a family $\equiv = \{\equiv_s\}_{s\in S}$ s.t.

1. $\equiv_s \subseteq T_\Sigma(X)_s \times T_\Sigma(X)_s$ for all $s \in S$; $(a, b) \in \equiv_s$ is denoted by $a \equiv_s b$;

2. $\equiv_s$ is symmetric and transitive, i.e. $t \equiv_s t'$ implies $t' \equiv_s t$ and $t \equiv_s t'$, $t' \equiv_s t''$ imply $t \equiv_s t''$ for all $t, t', t'' \in T_\Sigma(X)$. Denoting by $\mathrm{Dom}(\equiv_s)$ the set $\{t \mid t \equiv_s t\}$, define $t \equiv^D_s t'$ iff either $t \equiv_s t'$ or $t, t' \notin \mathrm{Dom}(\equiv_s)$;

3. $t_i {\equiv^D}_{s_i} t'_i$ for $i = 1 \dots n$ and $f \in F_{s_1 \dots s_n, s}$ imply $f(t_1, \dots, t_n) {\equiv^D}_s f(t'_1, \dots, t'_n)$;

4. $f(t_1, \dots, t_n) \in \mathrm{Dom}(\equiv)$, $t_i \notin \mathrm{Dom}(\equiv_{s_i})$ imply
$$f(t_1, \dots, t_{i-1}, x, t_{i+1}, \dots, t_n) \equiv_s f(t_1, \dots, t_{i-1}, t, t_{i+1}, \dots, t_n)$$
for all $t \in T_\Sigma(X)_s$ and all $x \in X_{s_i}$;

5. $x \notin \mathrm{Dom}(\equiv_s)$ for all $x \in X_s$.

Denoting by $[t]$ the equivalence class of t in $\equiv$, i.e. $\{t' \mid t \equiv t'\}$, the *quotient* $T_\Sigma(X)/\equiv$ is the non-strict algebra defined by:

- $s^{T_\Sigma(X)/\equiv}$ is $\{[t] \mid t \in \mathrm{Dom}(\equiv_s)\}$ for all $s \in S$;

- $f^{T_\Sigma(X)/\equiv}(\underline{t}) = [f(t_1, \dots, t_n)]$, where if $\underline{t}(i) \in s^{T_\Sigma(X)/\equiv}$ then $t_i \in \underline{t}(i)$ else $t_i \in X_{s_i}$, if $f(t_1, \dots, t_n) \in \mathrm{Dom}(\equiv_s)$, else $f^{T_\Sigma(X)/\equiv}(\underline{t})$ is undefined, for all $f \in F_{s_1 \dots s_n, s}$. □

Note that $f^{T_\Sigma(X)/\equiv}$ is well defined. Indeed let t_i, t'_i belong to $\underline{t}(i)$ for all i s.t. $\underline{t}(i) \in s^{T_\Sigma(X)/\equiv}$, otherwise t_i, t'_i belong to X_{s_i} and hence $t_i, t'_i \notin \mathrm{Dom}(\equiv_{s_i})$ because of 6; then $t_i \equiv^D t'_i$ for $i = 1 \dots n$ and hence, because of 3, $f(t_1, \dots, t_n) \equiv^D f(t'_1, \dots, t'_n)$ so that $[f(t_1, \dots, t_n)] = [f(t'_1, \dots, t'_n)]$.

As in more familiar frames, also in this case the evaluation of a term in a quotient algebra is the equivalence class of the term where variables have been replaced by (a representative of) their valuation.

Prop. 3.2.19 Let $\Sigma = (S, F)$ be a signature, $\equiv$ a congruence, Y an S-sorted family of variables, V a valuation for Y in $T_\Sigma(X)/\equiv$, U a substitution for $T_\Sigma(Y)$ in $T_\Sigma(X)$ s.t. $V(y) = [U(y)]$ for all $y \in Y$. Then $t^{T_\Sigma(X)/\equiv, V} = [U(t)]$ for any term t.
Proof. By induction over the structure of t. □

In order to apply the categorical construction of the initial object seen in the last section, the categorical structure of non-strict algebras is investigated, with a particular interest into two classical algebraic issues, the notion of *subalgebra* and *product*.

Def. 3.2.20 Let A be the non-strict algebra $(\{s^A\}_{s\in S}, \{f^A\}_{f\in F})$; the algebra B is a *weak subalgebra* of A iff

- $s^B \subseteq s^A$ for all $s \in S$;
- $f^B(\underline{b}) \leq f^A(\underline{b})$ for all $f \in F_{s_1 \ldots s_n, s}$ and all $\underline{b} \in {s_1}^B \times_p \ldots \times_p {s_n}^B$.

A weak subalgebra B of A is a *subalgebra* iff $f^B(\underline{b}) = f^A(\underline{b})$ for all $f \in F_{s_1 \ldots s_n, s}$ and all $\underline{b} \in {s_1}^B \times_p \ldots \times_p {s_n}^B$. □

It is easy to check, by induction on the structure of terms, that if B is a weak subalgebra of A and $V: X \to B$ is a valuation, then $t^{B,V} \leq t^{A, e\circ V}$ for any $t \in T_\Sigma(X)$, where $e: B \to A$ is the embedding and analogously that if B is a subalgebra of A, then $t^{B,V} = t^{A, e\circ V}$.

Weak subalgebras are categorical subobjects, i.e. are (up to isomorphism) the domains of monomorphisms, and subalgebras are regular subobjects, i.e. are (up to isomorphism) the domains of equalizers, as the following propositions state.

Prop. 3.2.21 Let $h: A_1 \to A_2$ be a homomorphism; then h is a monomorphism iff h_s is injective for all $s \in S$ iff A_1 is isomorphic to a weak subalgebra of A_2.
Proof. Assume that m is a monomorphism and show that m_s is injective for all $s \in S$. Assume by contradiction that there exists $\overline{s} \in S$ s.t. $m_{\overline{s}}$ is not injective, i.e. that there exist distinct $a, b \in \overline{s}^{A_1}$ s.t. $m_{\overline{s}}(a) = m_{\overline{s}}(b)$, and show that there exist two homomorphisms $g, h: C \to A_1$ s.t. $m \circ g = m \circ h$, but $g \neq h$.

Let C be the algebra defined by $s^C = \emptyset$ for all $s \neq \overline{s}$, $\overline{s}^C = \{x\}$ and f^C totally undefined for all $f \in F$.

Since f^C is totally undefined for all $f \in F$, every function from C into A_1 is a homomorphism; let g be defined by $g(x) = a$ and h by $h(x) = b$. Then $m \circ g = m \circ h$, but $g \neq h$, in contradiction with the assumption that m is a monomorphism.

It is immediate to check that if $m: A_1 \to A_2$ is injective, then A_1 is isomorphic to the weak subalgebra B of A_2, defined by:

Algebra $B =$
$$s^B = \{m(a) \mid a \in s^{A_1}\} \text{ for all } s \in S$$
$$f^B(m \circ \underline{a}) = m(f^{A_1}(\underline{a})) \text{ for all } f \in F$$

where f^B is well defined, because m is a total injective function, and $f^B \leq f^{A_2}$, because m is a homomorphism.

It remains to be shown that if A_1 is (isomorphic to) a weak subalgebra of A_2, then it is the domain of a monomorphism; let $m: A_1 \to A_2$ be the embedding and show that m is mono.

Let $g, h: C \to A_1$ be homomorphisms s.t. $m \circ g = m \circ h$; then $m(g(c)) = m(h(c))$ for any $c \in s^C$, i.e., as m is injective, $g(c) = h(c)$. Therefore $g = h$. $\square$

Prop. 3.2.22 Any two parallel homomorphisms $g, h: A_1 \to A_2$ have an equalizer $e: E(g,h) \to A_1$, the embedding of $E = E(g,h)$ into A_1, where E is defined by:

Algebra $E =$
$$s^E = \{a \mid a \in s^{A_1}, g(a) = h(a)\} \text{ for all } s \in S$$
$$f^E(\underline{a}) = f^{A_1}(\underline{a}) \text{ for all } f \in F \text{ and all } \underline{a} \in {s_1}^E \times_p \ldots \times_p {s_n}^E$$

Moreover a homomorphism $e: E \to A$ is an equalizer iff E is (isomorphic to) a subalgebra of A.

Proof. In order to show that such an $E(g,h)$ is the equalizer of g and h, first note that, by definition of homomorphism, $g(\underline{a}) = h(\underline{a})$ and $f^{A_1}(\underline{a}) \in s^{A_1}$ imply $g(f^{A_1}(\underline{a})) = h(f^{A_1}(\underline{a}))$, so that f^E is well defined.

Moreover the embedding e of E into A_1 obviously equalizes g and h. Thus it remains to be shown that any $m: C \to A_1$ s.t. $g \circ m = h \circ m$ factorizes in a unique way through e. Since $g \circ m = h \circ m$, $m(C) \subseteq E(f,g)$ and hence $m: C \to E(g,h)$ is the unique factorization of m through e.

Therefore if $e: E \to A_1$ is the equalizer of g and h, then, equalizers being unique up to isomorphism, E is isomorphic to the subalgebra $E(g,h)$ of A_1.

In order to show, on the converse, that any subalgebra is the domain of an equalizer, let B be a subalgebra of A and define C as follows:

- $s^C = \{(0,a), (1,a) \mid a \in s^A\}/\equiv$, where $(i,a) \equiv (i',a')$ iff $a = a'$ and ($i = i'$ or $a \in s^B$);
- for each $f \in F$ if there is $i \in \{0,1\}$ s.t. for every k the definedness of $\underline{c}(k)$ implies $\underline{c}(k) = [(i, a_k)]$, then $f^C(\underline{c}) = [(i, f^A(\underline{a}))]$, where $\underline{a}(k) = a_k$ for each k s.t. $\underline{c}(k) = [(i, a_k)]$ is defined and $\underline{a}(k)$ is undefined otherwise, else $f^C(\underline{c})$ is undefined.

It is immediate to check that $\equiv$ is an equivalence relation; thus in order to have that C is an algebra, it remains to be shown that f^C is well defined.

Let $\underline{c}$ be s.t. there are $i, i' \in \{0,1\}$ s.t. for every k the definedness of $\underline{c}(k)$ implies $\underline{c}(k) = [(i, a_k)] = [(i', a'_k)]$; since $(i, a_k) \equiv (i', a'_k)$, $a_k = a'_k$ and $i = i'$, so that $(i, f^A(\underline{a})) = (i', f^A(\underline{a}))$, or $a_k \in s^B$ for every k s.t. $\underline{a}(k)$ is defined, so that, since B is a subalgebra, $f^A(\underline{a})$ either is undefined or belongs to s^B and hence in both cases $(i, f^A(\underline{a})) \equiv^D (i', f^A(\underline{a}))$. Therefore f^C is well defined; moreover it is immediate to check that it is monotonic and hence C is an algebra; define $g, h\colon A \to C$ as follows and show that $B = E(g, h)$.

$$g(a) = [(0, a)] \text{ and } h(a) = [(1, a)] \text{ for all } a \in s^A.$$

Assume that $f^A(\underline{a})$ is defined; then $g(f^A(\underline{a})) = [(0, f^A(\underline{a}))] = f^C(\underline{c})$ for $\underline{c}$ defined by $\underline{c}(k) = [(0, \underline{a}(k))]$, i.e. for $\underline{c} = g \circ \underline{a}$ and hence g is a homomorphism. Thus $g(f^A(\underline{a})) = f^C(g \circ \underline{a})$. It is analogously easy to check that h is a homomorphism, too.

Finally $g(a) = h(a)$ iff $[(0, a)] = [(1, a)]$, i.e. iff $a \in s^B$; therefore $B = E(g, h)$. □

The product of non-strict algebras is defined in the usual way.

Def. 3.2.23 Let $\Sigma = (S, F)$ be a signature and D be a non-empty set of non-strict algebras over Σ. The *product* $\prod^{A \in D} A$ is the non-strict algebra P over Σ defined by:

- for all $s \in S$ let s^P be
$$\prod^{A \in D} s^A = \{g\colon D \to \cup_{A \in D} s^A \mid g(A) \in s^A \text{ for all } A \in D\};$$
- for all $f \in F_{s_1 \ldots s_n, s}$ let f^P be the function defined by: for every $\underline{p} \in {s_1}^P \times_p \ldots \times_p {s_n}^P$ $f^P(\underline{p})$ is defined iff $f^A(\underline{a})$ is defined for all $A \in D$, where $\underline{a}$ is defined by $\underline{a}(i) = \underline{p}(i)(A)$ for $i = 1 \ldots n$, and in this case $f^P(\underline{p})$ is defined by $f^P(\underline{p})(A) = f^A(\underline{a})$ for all $A \in D$.

The *projection* of $\prod^{A \in D} A$ into A, denoted by π^A, is the homomorphism defined by:
$$\pi^A(g) = g(A) \text{ for all } g \in s^{\prod^{A \in D} A} \text{ and all } s \in S.$$
In the sequel if D is the finite set $\{A_1, \ldots, A_n\}$, then $\prod^{A \in D} A$ is also denoted by $A_1 \times \ldots \times A_n$. □

Prop. 3.2.24 Non-strict weak subalgebras coincide with categorical subobjects, i.e. domain of monomorphisms, non-strict subalgebras coincide with categorical regular objects, i.e. domain of equalizers, and the product defined in Def. 3.2.23 coincides with the categorical product.

Proof. Because of Props. 3.2.21 and 3.2.22, B is a weak subalgebra iff it is the domain of a monomorphism, and it is a subalgebra iff it is the domain of an equalizer. Moreover it is trivial to check that the product defined in Def. 3.2.23 satisfies the universal property of the categorical product. □

The concept of *inductive* algebra, i.e. algebra satisfying the no-junk condition, is first introduced and then related to the idea of *term-generated*. Then it is shown that in every class of algebras closed w.r.t. inductive subalgebras the initial object, if any, is characterized by the no-junk and no-confusion properties.

Def. 3.2.25 Let A be a non-strict algebra; its *inductive part* $<A>$ is a family $\{s^{<A>}\}_{s \in S}$ of its carrier sub-sets inductively defined by:

$$\frac{f^A \in s^A}{f^A \in s^{<A>}}$$

for all $f \in F_{\Lambda,s}$

$$\frac{\underline{a} \in {s_1}^{<A>} \times_p \ldots \times_p {s_n}^{<A>}, f^A(\underline{a}) \in s^A}{f^A(\underline{a}) \in s^{<A>}}$$

for all $f \in F_{s_1 \ldots s_n, s}$

The *inductive subalgebra* B of A consists of:

- $s^B = s^{<A>}$ for all $s \in S$;
- $f^B(\underline{b}) = f^A(\underline{b})$ for all $f \in F_{s_1 \ldots s_n, s}$ and all $\underline{b} \in \times_p\{{s_1}^{<A>}, \ldots, {s_n}^{<A>}\}$.

In the sequel the inductive subalgebra of an algebra A will be denoted by $<A>$.

A non-strict algebra A is *inductive* iff $A = <A>$.

The *embedding* of $<A>$ into A is the homomorphism $e = \{e_s\}_{s \in S}$ defined by $e_s(a) =_e a$ for all $a \in s^{<A>}$.

For any non-strict algebra A let $\equiv^A$ be the congruence defined by $t \equiv^A t'$ iff $t^{A,V_?} =_e t'^{A,V_?}$ and $i^A \colon T_\Sigma(X)/\equiv^A \rightarrow <A>$ be the isomorphism defined by $i^A([t]) = t^{A,V_?}$.

Let C be a class of non-strict algebras on a signature Σ; the subclass $\mathsf{Ind}(C)$ of C consists of $\{A \mid A \in C, A \text{ is inductive}\}$. □

Note that the definition of $s^{<A>}$ guarantees both the well definedness of $f^{<A>}$ and the embedding being a homomorphism.

The usual equivalence between inductive and term-generated algebras has to be a little relaxed, because functions over terms are total, while in inductive algebras may also be non-strict. Thus some syntactic elements are needed to play the role of the "undefined" elements which cooperate to build the carriers; for that a family X of variables with the totally undefined valuation $V_?$ over them is used.

Prop. 3.2.26 Let $\Sigma = (S, F)$ be a signature, X be any family $\{X_s\}_{s \in S}$ of variables s.t. $X_s \neq \emptyset$ for all $s \in S$ and A a non-strict algebra; the following conditions are equivalent.

1. A is inductive;
2. $eval^{A,V_?}\colon T_\Sigma(X) \to A$ is surjective;
3. for any algebra B there exists at most one homomorphism $k\colon A \to B$;
4. A has no proper subalgebras.

Proof.

$1 \Rightarrow 2$ Since A is inductive, $s^A = s^{<A>}$ and hence we show by induction that for all

$a \in s^{<A>}$ there exists $t \in T_\Sigma(X)$ s.t. $t^{A,V_?} = a$.

If $a = f^A$ for some $f \in F_{\Lambda,s}$, then $f \in T_\Sigma(X)_s$ by definition of term algebra and $a = f^{A,V_?}$.

Otherwise $a = f^A(\underline{a})$ for some $f \in F_{s_1 \ldots s_n, s}$ and $\underline{a} \in {s_1}^{<A>} \times_p \ldots \times_p {s_n}^{<A>}$; because of inductive hypothesis for each i s.t. $\underline{a}(i) \in {s_i}^{<A>}$ there exists $t_i \in T_\Sigma(X)_{s_i}$ s.t. $\underline{a}(i) = {t_i}^{A,V_?}$. For all i s.t. $\underline{a}(i) \notin {s_i}^{<A>}$ let t_i be any element of X_{s_i}, which exists because X_s is non-empty for all $s \in S$. Then $\underline{a}(i) = {t_i}^{A,V_?}$ for all $i = 1 \ldots n$, by definition of t_i and of $V_?$, and hence $f(t_1, \ldots, t_n)^{A,V_?} = f^A(\underline{a})$, i.e. $f(t_1, \ldots, t_n)^{A,V_?} = a$.

$2 \Rightarrow 3$ Let $h, k\colon A \to B$ be homomorphisms; by hypothesis for each $a \in s^A$ there exists $t \in T_\Sigma(X)$ s.t. $t^{A,V_?} {=_e} a$; moreover $t^{A,V_?} \in s^A$ implies $h(t^{A,V_?}) {=_e} t^{B,V_?} {=_e} k(t^{A,V_?})$, because of Prop. 3.2.17. Therefore $h(a) {=_e} k(a)$ for all $a \in s^A$ and hence $h = k$.

$3 \Rightarrow 4$ Assume that E is a subalgebra of A; then there exist $g, h: A \to B$ s.t. E is their equalizer, because of Prop. 3.2.21. By hypothesis $g = h$ and hence, by construction of equalizer, $E = A$. Thus A has no proper subalgebras.

$4 \Rightarrow 1$ Since A has no proper subalgebras and $<A>$ is a subalgebra of A, $A = <A>$. $\square$

Prop. 3.2.27 Let $\Sigma = (S, F)$ be a signature, X any family $\{X_s\}_{s\in S}$ of variables s.t. X_s is non-empty for all $s \in S$ and C be a class of non-strict algebras over Σ closed w.r.t. inductive subalgebras, i.e. s.t. $A \in C$ implies $<A> \in C$.

A non-strict algebra $I \in C$ is initial in C iff it satisfies the following two conditions

1. I is inductive (*no-junk*);

2. $t^{I,V_?} =_e t'^{I,V_?}$ implies $t^{A,V_?} =_e t'^{A,V_?}$ for all $A \in C$ and all $t, t' \in T_\Sigma(X)$ (*no-confusion*).

Moreover I is initial in C iff it is initial in $\mathsf{Ind}(C)$.

Proof. I is initial in C iff it satisfies conditions 1 and 2.

$\Rightarrow$ Let I be initial in C; then $I \in C$ and hence, since C is closed w.r.t. inductive subalgebras, $<I> \in C$, too. Thus there exists one morphism $h: I \to <I>$, because I is initial. Let e denote the embedding of $<I>$ into I; since e is a homomorphism, $e \circ h$ is a homomorphism too and hence is the identity. Therefore, by definition of e, h is the identity too and hence $I = <I>$, i.e. I satisfies 1.

Assume that $t^{I,V_?} =_e t'^{I,V_?}$ for certain $t, t' \in T_\Sigma(X)$; then for each $A \in C$, because of Prop. 3.2.17 for h the unique homomorphism from I into A and $V = V_? = V'$, $t^{A,V_?} =_e t'^{A,V_?}$.

$\Leftarrow$ Let I satisfy conditions 1 and 2 and $h^A: I \to A$ be defined by $h^A(t^{I,V_?}) = t^{A,V_?}$ for all $t \in T_\Sigma(X)$ and all $A \in C$; then h^A is a well defined total function from $eval^{I,V_?}(T_\Sigma(X))$ into A, because of condition 2. Thus, $eval^{I,V_?}$ being surjective because of Prop. 3.2.26 and condition 1, h^A is a well defined total function from I into A. Finally h^A is a homomorphism by definition and it is unique, because of Prop. 3.2.26 and condition 1.

I is initial in C iff it is initial in $\mathsf{Ind}(C)$.

$\Rightarrow$ Let I be initial in C; then I satisfies condition 1 and hence $I \in \mathsf{Ind}(C)$. Since I is initial in C, for all $A \in \mathsf{Ind}(C) \subseteq C$ there exists exactly one homomorphism from I into A; thus I is initial in $\mathsf{Ind}(C)$.

$\Leftarrow$ Let I be initial in $\mathsf{Ind}(C)$ and A belong to C; then $<A> \in C$ and hence $<A> \in \mathsf{Ind}(C)$. Therefore there exists one morphism $h: I \to <A>$. Thus the composition $e \circ h$ of h with the embedding e of $<A>$ into A is a homomorphism and it is unique, because of Prop. 3.2.26 and I being inductive. Therefore for all $A \in C$ there exists exactly one homomorphism from I into A, i.e. I is initial in C. □

To check the existence of an initial model in a class closed w.r.t. inductive subalgebras, because of the above Prop. 3.2.27, it is sufficient to work on the subclass of inductive models; this is a real simplification, because the (isomorphism classes of) algebras form a proper class, while the subclass of the (isomorphism classes of) inductive algebras is a set. Thus the syntactical characterization of the initial model suggested by Prop. 3.2.27, as a quotient of a term algebra w.r.t. the intersection of the kernels of the natural evaluation of terms in *all* models, does not introduce foundational problems, because it is possible to work on the *set* of (canonical representatives for the isomorphism classes of) inductive models.

Def. 3.2.28 Let C be a non-empty class of non-strict algebras over Σ and D be the set of non-strict algebras defined by $D = \{T_\Sigma(X)/\equiv^A \mid A \in C\}$. Then $I(C)$ denotes the inductive sub-algebra of the product $\prod^{B \in D} B$. □

Theorem 3.2.29 Let $\Sigma = (S, F)$ be a signature and C be a class of non-strict algebras over Σ closed under isomorphisms and inductive subalgebras. The following conditions are equivalent:

1. there exists an initial object in C;
2. there exists an initial object in $\mathsf{Ind}(C)$;
3. $I(C)$ belongs to C;
4. $I(C)$ is initial in C.

Proof.

$1 \Leftrightarrow 2$ Because of Prop. 3.2.27.

$2 \Rightarrow 3$ Let I be initial in $\mathsf{Ind}(C)$; since C is closed under isomorphism, it is sufficient to show that I is isomorphic to $I(C)$.

Since C is closed under inductive subobjects and isomorphisms, for any $A \in C$ the algebra $T_\Sigma(X)/\equiv^A \in C$ and hence, as I is initial in $\mathsf{Ind}(C)$, there exists (a unique) $h^A: I \to T_\Sigma(X)/\equiv^A$. Therefore, by definition of product in a categorical setting, there exists a morphism $h: I \to \prod^{B \in D} B$, where D is the set $\{T_\Sigma(X)/\equiv^A \mid A \in C\}$.

Since I is inductive, Prop. 3.2.26 implies that such a h is unique and, by construction, $h: I \to <\prod^{B \in D} B>$ i.e. $h: I \to I(C)$. Since both I and $I(C)$ are inductive, to show that h is an isomorphism it is sufficient to show that there exists a homomorphism $k: I(C) \to I$; indeed $h \circ k$ should be the unique homomorphism from $I(C)$ into itself, i.e. the identity, and analogously for $k \circ h$.

Consider the composition of the following homomorphisms:

- the embedding $e: I(C) \to \prod^{B \in D} B$;
- the projection $\pi^I: \prod^{B \in D} B \to T_\Sigma(X)/\equiv^I$;
- the isomorphism $i^I: T_\Sigma(X)/\equiv^I \to <I>$;
- the embedding $e_I: <I> \to I$;

and get the thesis for $k = e_I \circ i^I \circ \pi^I \circ e$.

$3 \Rightarrow 4$ Let $I(C)$ belong to C and A be any element of C; since $I(C)$ is inductive, there exists at most one homomorphism from I into A, because of Prop. 3.2.26, so that it is sufficient to prove that there exists a homomorphism h^A from $I(C)$ into A.

To define such a homomorphism consider the composition of the following homomorphisms:

- the embedding $e: I(C) \to \prod^{B \in D} B$;
- the projection $\pi^A: \prod^{B \in D} B \to T_\Sigma(X)/\equiv^A$;
- the isomorphism $i^A: T_\Sigma(X)/\equiv^A \to <A>$;
- the embedding $e^A: <A> \to A$;

and get the thesis for $h^A = e_A \circ i^A \circ \pi^A \circ e$.

$4 \Rightarrow 1$ Trivial. □

Non-strict Theories

Usually in both the total and the partial frame, logical formulas (equations and positive Horn clauses) are considered s.t. their model classes are non-empty and closed w.r.t. non-empty products and sub-objects, so that the model classes satisfy *a fortiori* the closure w.r.t.

$I(C)$ which is necessary and sufficient for the existence of an initial model, by Theorem 3.2.29, for classes closed under subobjects and isomorphisms. In the non-strict frame, the same way cannot be followed, because there are finite sets of equations whose model classes are neither closed w.r.t. $I(C)$, nor w.r.t. non-empty products. This claim is informally shown by a simple example.

spec sp_7 =
  sorts s
  opns
    $k, k' \colon \to s$
    $f \colon s \to s$
  axioms
    $f(k) =_e k'$

The following two algebras are obviously models of sp_7:

Algebra A =
  $s^A = \{\cdot\}$
  $k^A = k'^A = \cdot$
  f^A is the total strict function defined by $f^A(\cdot) = \cdot$

Algebra B =
  $s^B = \{\cdot\}$
  $k'^B = \cdot$ and k^B is undefined
  $f^B(\underline{b}) = \cdot$ for all $\underline{b} \in \times_p s^B$

Let C be the model class of sp_7. By definition of product, both $k^{A\times B}$ and $k^{I(C)}$ are undefined, because k^B is undefined, and analogously both $f^{A\times B}(g)$ and $f^{I(C)}(g)$, where g is the totally undefined function, are undefined too,

because f^A is strict. Therefore both $f(k)^{A\times B}$ and $f(k)^{I(C)}$ are undefined and hence both $A \times B$ and $I(C)$ are not models of sp_7.

In the above example the problem arises because of the monotonicity of the interpretation of the function symbols; indeed from $f(a) =_e b$ in each model A either $a =_e a$ or $f(x) =_e b$ holds. Thus equations implicitly introduce disjunctions. Moreover, using conditional axioms, it is possible to code each disjunction and hence in the non-strict frame it is equivalent to deal with equations or disjunctions. Consider the following informal proof of this claim.

Let $\epsilon_1 \vee \ldots \vee \epsilon_n \vee \neg\eta_1 \vee \ldots \vee \neg\eta_m$, where

$\epsilon_1, \ldots, \epsilon_n, \eta_1, \ldots, \eta_m$ are all existential equalities; then $\epsilon_1 \vee \ldots \vee \epsilon_n \vee \neg\eta_1 \vee \ldots \vee \neg\eta_m$ may be coded by the set:

$$\alpha_i \qquad D(f_i(x)) \wedge \eta_1 \wedge \ldots \wedge \eta_m \supset \epsilon_i$$

for $i = 1 \ldots n$ and

$$\alpha \qquad D(f_n(f_{n-1}(\ldots(x)\ldots)$$

where $f_1, \ldots, f_n$ are auxiliary unary functions. Indeed each algebra A satisfying α satisfies also at least one $D(f_i(x))$ and hence if A satisfies also α_i either there exists an η_j s.t. A does not satisfy η_j or A satisfies ϵ_i, so that A satisfies $\epsilon_1 \vee \ldots \vee \epsilon_n \vee \neg\eta_1 \vee \ldots \vee \neg\eta_m$ Vice versa if $\epsilon_1 \vee \ldots \vee \epsilon_n \vee \neg\eta_1 \vee \ldots \vee \neg\eta_m$ holds for A, then A may generalize to a model of $\alpha_1, \ldots, \alpha_n, \alpha$, suitably defining the interpretation of $f_1, \ldots, f_n$. Therefore in the sequel the focus is on *disjunctive* specifications.

Def. 3.2.30 Let $\Sigma = (S, F)$ be a signature and X be a family of S-sorted variables.

- The set $Eq(\Sigma, X)$ of *equalities* on Σ and X consists of $t =_e t'$ for all $t, t' \in T_\Sigma(X)_s$, $s \in S$; the set $At(\Sigma, X)$ of *atomic* formulas on Σ and X is $Eq(\Sigma, X) \cup \{\neg\epsilon \mid \epsilon \in Eq(\Sigma, X)\}$.

- The set $Form(\Sigma, X)$ of all *well-formed* formulas is inductively defined by:

  - $Eq(\Sigma, X) \subseteq Form(\Sigma, X)$.
  - $\Phi \cup \{\theta, \theta'\} \subseteq Form(\Sigma, X)$ implies $\wedge\Phi, \vee\Phi, \neg\theta, \theta \supset \theta' \in Form(\Sigma, X)$.

  For every well-formed formula ϕ, $Var(\phi)$ denotes the set of variables which appear in ϕ.

- The set $Cond(\Sigma, X)$ of *conditional formulas* on Σ and X is the set $\{\wedge\Delta \supset \epsilon \mid \Delta \cup \{\epsilon\} \subseteq Eq(\Sigma, X)\}$. If Δ is the empty set, then $\wedge\Delta \supset \epsilon$ is an equivalent notation for ϵ and hence $Eq(\Sigma, X) \subseteq Cond(\Sigma, X)$.

- The set $DForm(\Sigma, X)$ of *disjunctive formulas* on Σ and X is the set $\{\vee\Delta \mid \Delta \subseteq At(\Sigma, X)\}$. If Δ consists of one atomic formula ϵ, then $\vee\Delta$ is an equivalent notation for ϵ and hence $At(\Sigma, X) \subseteq DForm(\Sigma, X)$. □

Def. 3.2.31 Let $\Sigma = (S, F)$ be a signature, X be a family of S-sorted variables and A be a non-strict Σ-algebra.

If ϕ is a formula and V is a valuation for $Var(\phi)$ in A, then ϕ *holds for* V *in* A (equivalently: *is satisfied for* V *by* A), denoted by $A \models_{\mathcal{DNS}_V} \phi$, accordingly to the following definitions.

- $A \models_{\mathcal{DNS}_V} t =_e t'$ iff $t^{A,V} t'^{A,V} \in s^A$ and $t^{A,V} = t'^{A,V}$;
- $A \models_{\mathcal{DNS}_V} \wedge\, \Phi$ iff $A \models_{\mathcal{DNS}_V} \phi$ for all $\phi \in \Phi$;
- $A \models_{\mathcal{DNS}_V} \vee\, \Phi$ iff there exists $\phi \in F$ s.t. $A \models_{\mathcal{DNS}_V} \phi$;
- $A \models_{\mathcal{DNS}_V} \neg\theta$ iff $A \not\models_{\mathcal{DNS}_V} \theta$;
- $A \models_{\mathcal{DNS}_V} \theta \supset \theta'$ iff $A \models_{\mathcal{DNS}_V} \theta'$ or $A \not\models_{\mathcal{DNS}_V} \theta$.

A formula ϕ *holds in* (equivalently: *is satisfied by, is valid in*) A, denoted by $A \models_{\mathcal{DNS}} \phi$, iff $A \models_{\mathcal{DNS}_V} \phi$ for all valuations V for $Var(\phi)$ in A. Let $D(t)$ shortly denote the equality $t =_e t$, where both sides are the same term, because $t =_e t$ simply states the definedness of t. □

Note that $A \models_{\mathcal{DNS}_V} \wedge\, \emptyset$ for all non-strict algebras A and all valuations V because obviously $A \models_{\mathcal{DNS}_V} \phi$ for all $\phi \in \emptyset$, so that $\wedge\emptyset$ plays the role of the constant *True*, and $A \not\models_{\mathcal{DNS}_V} \vee\, \emptyset$ for all non-strict algebras A and all valuations V because obviously there does not exist $\phi \in \emptyset$ s.t. $A \models_{\mathcal{DNS}_V} \phi$, so that $\vee\emptyset$ plays the role of the constant *False*.

Remark. In both the total and the partial frame, since valuations are *total* functions, the relation $\equiv$ on $T_\Sigma(X)$, defined by $t \equiv t'$ iff $A \models t =_e t'$, may be not an equivalence relation if empty carriers are allowed, because it may be not transitive, as it has been shown in Sect. 2.4.2 and this fact has consequences in the case of inference systems, which have to deal very carefully with the elimination of the variables.

On the contrary, in the non-strict frame these problems do not arise, because valuations are *partial* functions, so that there exists at least the totally undefined valuation for all families of variables and all non-strict algebras. Moreover the following Prop. 3.2.32 shows that $\equiv$ coincides with the relation $\equiv_?$, defined by $t \equiv_? t'$ iff $A \models_{\mathcal{DNS}_{V_?}} t =_e t'$, and hence is an equivalence relation for any non-strict Σ-algebra A.

Manca and Salibra in [58] introduced the partial valuations to solve the empty-carriers problem and keep the original Birkhoff equational calculus, by changing the concept of validity; but note that here the introduction of partial valuations has a completely different flavor; indeed it is not a technical device as in [58], but it arises naturally from the setting, since functions are non-strict and variables have to represent *all* the possible arguments. $\square$

Prop. 3.2.32 Let $\Sigma = (S, F)$ be a signature, X be a family of S-sorted variables, Δ be a set of equalities over Σ and X.

Then $A \models_{\mathcal{DNS}} t =_e t'$ iff $A \models_{\mathcal{DNS}_{V_?}} t =_e t'$ for all terms t and t' for every non-strict Σ-algebra A and moreover the following conditions are equivalent:

1. $A \models_{\mathcal{DNS}} \vee \Delta$;
2. $A \models_{\mathcal{DNS}_{V_?}} \vee \Delta$;
3. there exists $\delta \in \Delta$ s.t. $A \models_{\mathcal{DNS}} \delta$;

Proof. It is first shown that $A \models_{\mathcal{DNS}} t =_e t'$ iff $A \models_{\mathcal{DNS}_{V_?}} t =_e t'$.

$\Rightarrow$ If $A \models_{\mathcal{DNS}} t =_e t'$, then, by definition of validity, $A \models_{\mathcal{DNS}_V} t =_e t'$ for all valuations V, so that in particular $A \models_{\mathcal{DNS}_{V_?}} t =_e t'$.

$\Leftarrow$ Since $A \models_{\mathcal{DNS}_{V_?}} t =_e t'$, $t^{A,V_?}, t'^{A,V_?} \in s^A$ and $V_? \leq V$ for all valuation V by definition of $V_?$; thus, by Prop. 3.2.16, $t^{A,V} =_e t^{A,V_?}$ and $t'^{A,V} =_e t'^{A,V_?}$. Therefore from $t^{A,V_?} =_e t'^{A,V_?}$, $t^{A,V} =_e t'^{A,V}$ follows and hence $A \models_{\mathcal{DNS}_{V_?}} t =_e t'$.

The above conditions are now shown to be equivalent.

$1 \Rightarrow 2$ If $A \models_{\mathcal{DNS}} \vee \Delta$, then, by definition of validity, $A \models_{\mathcal{DNS}_V} \vee \Delta$ for all valuations V, so that in particular $A \models_{\mathcal{DNS}_{V_?}} \vee \Delta$.

$2 \Rightarrow 3$ Assume that $A \models_{\mathcal{DNS}_{V_?}} \vee \Delta$; then, by definition of validity, there exists $t =_e t' \in \Delta$ s.t. $A \models_{\mathcal{DNS}_{V_?}} t =_e t'$; thus, since it has already be shown that $A \models_{\mathcal{DNS}} t =_e t'$ iff $A \models_{\mathcal{DNS}_{V_?}} t =_e t'$, $A \models_{\mathcal{DNS}} t =_e t'$.

$3 \Rightarrow 1$ Trivial. $\square$

Note that if Δ is a set of atoms, i.e. of equalities and negated equalities, then $A \models_{\mathcal{DNS}_{V_?}} \vee \Delta$ does not imply $A \models_{\mathcal{DNS}} \vee \Delta$. For example if Δ is $\{\neg D(x)\}$, then $\vee \Delta$ is satisfied for the totally undefined valuation, while each non-empty algebra does not satisfy it.

Def. 3.2.33 A *specification sp* consists of a signature Σ and of a set of well-formed formulas over Σ, called *axioms* of *sp*.

A specification is called respectively *disjunctive, conditional, equational*, if all the axioms are disjunctions, conditional formulas, equalities, respectively.

Let $sp = (\Sigma, Ax)$ be a specification; the class $\mathrm{NSMod}(sp)$ of *models* of sp is

$$\{A \mid A \in \mathrm{NSAlg}(\Sigma), A \models_{\mathcal{DNS}} \alpha \text{ for all } \alpha \in Ax\}.$$

A model of sp is *initial* for sp iff it is initial in $\mathrm{NSMod}(sp)$. □

Remark. Note that disjunctive specifications are sufficient to define any class of models definable using well-formed formulas. Indeed each well-formed formula over the usual logical connectives may be expressed in conjunctive normal form, as it is possible to prove directly in the non-strict frame following the same pattern of the proof in first-order logic. Since a conjunction of formulas is logically equivalent to the set of the formulas in the conjunction, each well-formed formula (in conjunctive normal form) is logically equivalent to a set of disjunctive formulas.

The expressive power of equalities in a non-strict frame is quite different from the usual one; for instance $\mathrm{NSMod}(sp)$ may be empty also for equational specifications. For example if $D(x)$ is an axiom of the specification, then no algebra can satisfy this axiom w.r.t. the valuation completely undefined, so that the specification has no models.

Def. 3.2.34 A specification sp is *consistent* iff $\mathrm{NSMod}(sp)$ is not empty. □

Although the class of models of a disjunctive specification is not a variety, because it is not closed under products nor quotients, it is at least closed under (inductive) subalgebras and isomorphisms and these closures are sufficient to instantiate Prop. 3.2.27 and Theorem 3.2.29.

Prop. 3.2.35 The class of models of a disjunctive specification is closed w.r.t. subalgebras and isomorphisms.

Proof. Let $sp = (\Sigma, Ax)$ be a disjunctive specifications and C denote $\mathrm{NSMod}(sp)$. By definition of validity, if A and B are isomorphic non-strict algebras, then $A \models_{\mathcal{DNS}} \phi$ iff $B \models_{\mathcal{DNS}} \phi$ for all disjunctive formulas ϕ; thus C is closed w.r.t. isomorphisms.

Let B be a subalgebra of A, for some $A \in C$, and show that $B \in C$. Let α belong to Ax and V be a valuation for the variables of α in B; then it is also a valuation for the variables of α in A and hence $A \models_{\mathcal{DNS}_V} \alpha$, because $A \in \mathrm{NSMod}(sp)$. Since B is a subalgebra of A, $t^{A,V} = t^{B,V}$ for all terms t and hence, by definition of validity, $A \models_{\mathcal{DNS}_V} \epsilon$ iff $B \models_{\mathcal{DNS}_V} \epsilon$ for all equalities ϵ. Thus, α being a disjunctive formula, $A \models_{\mathcal{DNS}_V} \phi$ implies $B \models_{\mathcal{DNS}_V} \phi$. □

Note that model classes of disjunctive specifications may be non-closed w.r.t. weak subalgebras; indeed consider for example the axiom $D(a)$, which simply states the definedness of a constant a; then in any model A of $D(a)$ the constant a denotes an element a^A of the carrier of A, but a weak subalgebra B of A may exist s.t. a^B is undefined, so that B is not a model of $D(a)$.

Theorem 3.2.36 Let $\Sigma = (S, F)$ be a signature and $sp = (\Sigma, Ax)$ be a disjunctive specification. The following conditions are equivalent:

1. there exists an initial model in $\mathrm{NSMod}(sp)$;
2. there exists an initial model in $\mathsf{Ind}(\mathrm{NSMod}(sp))$;
3. $I(\mathrm{NSMod}(sp)) \in \mathrm{NSMod}(sp)$;
4. $I(\mathrm{NSMod}(sp))$ is initial in $\mathrm{NSMod}(sp)$.

Moreover a non-strict algebra $I \in \mathrm{NSMod}(sp)$ is initial in $\mathrm{NSMod}(sp)$ iff it is initial in $\mathsf{Ind}(\mathrm{NSMod}(sp))$ iff it is isomorphic to $I(\mathrm{NSMod}(sp))$ iff it satisfies the following two conditions

a I is inductive;

b $I \models_{\mathcal{DNS}} t =_e t'$ implies $A \models_{\mathcal{DNS}} t =_e t'$ for all $A \in \mathrm{NSMod}(sp)$ and all $t, t' \in T_\Sigma(X)$, where X is an S-sorted family of variables s.t. X_s is non-empty for all $s \in S$.

Proof. Because of Prop. 3.2.35, $\mathrm{NSMod}(sp)$ is closed w.r.t. inductive subalgebras and isomorphisms, so that Theorem 3.2.29 and Prop. 3.2.27 apply. $\Box$

Fact 3.2.37 Equational consistent specifications may do not have an initial model.

Proof.

spec sp_8 =
  sorts s
  opns
    $f, g \colon s \to s$
  axioms
    $D(g(f(x)))$

Then sp_8 is a consistent equational specification, because the non-strict algebra A, defined as follows, is a model of sp_8.

Algebra $A =$
$s^A = \{\cdot\}$
$f^A(\underline{a}) = g^A(\underline{a}) = \cdot$ for all $\underline{a} \in \times_p\{s^A\}$

There are two models A and B of sp_8, defined below, s.t. respectively $f^A(?) \notin s^A$ and $g^B(?) \notin s^B$ and hence for any algebra I satisfying the condition 3.2.36 of Theorem 3.2.36 $g^I(f^I(?)) \notin s^I$ so that it is not a model of sp_8; thus sp_8 has no initial model, because of Theorem 3.2.36.

Algebra $A =$
$s^A = \{\cdot\}$
f^A is totally undefined
$g^A(\underline{a}) = \cdot$ for all $\underline{a} \in \times_p\{s^A\}$

Algebra $B =$
$s^B = \{\cdot\}$
$f^B(\underline{b}) = \cdot$ for all $\underline{b} \in \times_p\{s^B\}$
g^B is totally undefined

To give necessary and sufficient conditions for the existence of the initial model, some preliminary results are needed.

Variables play the role of the "undefined" objects and hence, because of monotonicity, they may be replaced by any other term in any formula without affecting its validity. Moreover this replacement may be also "asymmetric", changing different occurrences of the same variable in a formula by different terms; for example from $A \models_{\mathcal{DNS}} f(x) =_e f'(x)$, for the valuation $V_?$ in an algebra A, $f^A(?) =_e f'^A(?)$ and hence, by monotonicity, $f^A(a) =_e f'^A(b)$ for all $a, b \in A$; thus $A \models_{\mathcal{DNS}} f(x) =_e f'(y)$.

Lemma 3.2.38 Let A be an algebra over a signature $\Sigma = (S, F)$; then $A \models_{\mathcal{DNS}} D(f(t_1, \ldots, t_n))$ implies

$$A \models_{\mathcal{DNS}} f(t_1, \ldots, t_n) =_e f(t_1, \ldots, t_{i-1}, x, t_{i+1}, \ldots, t_n) \vee D(t_i),$$

for every $i = 1 \ldots n$.

Proof. Assume that $A \models_{\mathcal{DNS}} D(f(t_1, \ldots, t_n))$; then in particular $A \models_{\mathcal{DNS}_{V_?}} D(f(t_1, \ldots, t_n))$ so that either $A \models_{\mathcal{DNS}_{V_?}} D(t_i)$ or $t_i^{A,V_?} = x^{A,V_?}$ and hence

$$A \models_{\mathcal{DNS}_{V_?}} f(t_1, \ldots, t_n) =_e f(t_1, \ldots, t_{i-1}, x, t_{i+1}, \ldots, t_n).$$

Therefore

$$A \models_{\mathcal{DNS}_{V_?}} D(t_i) \vee f(t_1, \ldots, t_n) =_e f(t_1, \ldots, t_{i-1}, x, t_{i+1}, \ldots, t_n)$$

and hence, because of Prop. 3.2.32,

$$A \models_{\mathcal{DNS}} D(t_i) \vee f(t_1, \ldots, t_n) =_e f(t_1, \ldots, t_{i-1}, x, t_{i+1}, \ldots, t_n). \quad \Box$$

The initial model of a disjunctive specification exists iff any disjunction of *non-negated* atoms which is valid in all models has a privileged element holding in all models; thus disjunctions, which potentially cause troubles, may be solved and replaced by atoms. Moreover it is sufficient to check the property for just two kinds of disjunctions:

1. the disjunctions implicitly introduced because of monotonicity (see Lemma 3.2.38);
2. the disjunctions coming from instantiations of proper axioms.

Theorem 3.2.39 Let $sp = (\Sigma, Ax)$ be a consistent disjunctive specification. The following conditions are equivalent:

1. there exists I initial in $\mathrm{NSMod}(sp)$.
2. there exists I initial in $\mathsf{Ind}(\mathrm{NSMod}(sp))$.
3. $I(\mathrm{NSMod}(sp)) \in \mathrm{NSMod}(sp)$.
4. $I(\mathrm{NSMod}(sp))$ is initial in $\mathrm{NSMod}(sp)$.
5. for all sets Δ of equalities

$$A \models_{\mathcal{DNS}} \vee \Delta \text{ for all } A \in \mathrm{NSMod}(sp)$$

implies

$$\text{there exists } \delta \in \Delta s.t.\ A \models_{\mathcal{DNS}} \delta \text{ for all } A \in \mathrm{NSMod}(sp)$$

6. (a) for all $f \in F_{s_1 \dots s_n, s}$

$$A \models_{\mathcal{DNS}} D(f(t_1, \dots, t_n)) \text{ for all } A \in \mathrm{NSMod}(sp)$$

implies that (at least) one between the following properties holds

- $A \models_{\mathcal{DNS}} D(t_i)$ for all $A \in \mathrm{NSMod}(sp)$
- $A \models_{\mathcal{DNS}} f(t_1, \dots, t_n) =_e f(t_1, \dots, t_{i-1}, x, t_{i+1}, \dots, t_n)$ for all $A \in \mathrm{NSMod}(sp)$

(b) for all $\vee\Delta \in Ax$ and all substitutions $U \colon T_\Sigma(Var(\vee\Delta)) \to T_\Sigma(X)$

$$A \models_{\mathcal{DNS}} t =_e t' \text{ for all } A \in \mathrm{NSMod}(sp) \text{ and all } \neg t =_e t' \in U(\Delta)$$

implies

$$\text{there exists } \delta \in U(\Delta) \cap Eq(\Sigma, X) s.t.\ A \models_{\mathcal{DNS}} \delta \text{ for all } A \in \mathrm{NSMod}(sp)$$

7. the relation $\equiv$ over $T_\Sigma(X)$, defined by $t \equiv t'$ iff $A \models_{\mathcal{DNS}} t =_e t'$ for all $A \in$ NSMod(sp), is a congruence and $T_\Sigma(X)/\equiv$ is a model.

8. the relation $\equiv$ over $T_\Sigma(X)$, defined by $t \equiv t'$ iff $A \models_{\mathcal{DNS}} t =_e t'$ for all $A \in$ NSMod(sp), is a congruence and $T_\Sigma(X)/\equiv$ is the initial model.

Proof.

$1 \Leftrightarrow 2$ Because of Theorem 3.2.36.

$2 \Leftrightarrow 3$ Because of Theorem 3.2.36.

$3 \Leftrightarrow 4$ Because of Theorem 3.2.36.

$4 \Rightarrow 5$ Let $I = I(\text{NSMod}(sp))$ be the initial model and Δ be a set of equalities; thus

$A \models_{\mathcal{DNS}} \vee \Delta$ for all $A \in$ NSMod(sp) implies in particular $I \models_{\mathcal{DNS}} \vee \Delta$. Because of Prop. 3.2.32, $I \models_{\mathcal{DNS}} \vee \Delta$ implies that there exists $t =_e t' \in \Delta$ s.t. $I \models_{\mathcal{DNS}} t =_e t'$; then, because of Theorem 3.2.36, $A \models_{\mathcal{DNS}} t =_e t'$ for all $A \in$ NSMod(sp).

$5 \Rightarrow 6$ Assume that $A \models_{\mathcal{DNS}} D(f(t_1, \ldots, t_n))$ for all $A \in$ NSMod(sp).

Then $A \models_{\mathcal{DNS}} D(t_i) \vee f(t_1, \ldots, t_n) =_e f(t_1, \ldots, t_{i-1}, x, t_{i+1}, \ldots, t_n)$ for all $A \in$ NSMod(sp), by Lemma 3.2.38, and hence

$$A \models_{\mathcal{DNS}} f(t_1, \ldots, t_n) =_e f(t_1, \ldots, t_{i-1}, x, t_{i+1}, \ldots, t_n)$$

for all $A \in$ NSMod(sp) or $A \models_{\mathcal{DNS}} D(t_i)$ for all $A \in$ NSMod(sp), because of condition 5; thus 6a holds.

Assume that there exists $\vee\Delta \in Ax$ and $U \colon T_\Sigma(Var(\vee\Delta)) \to T_\Sigma(X)$ s.t. $A \models_{\mathcal{DNS}} t =_e t'$ for all $A \in$ NSMod(sp) and all $\neg t =_e t' \in U(\Delta)$. Then, since both $A \models_{\mathcal{DNS}} t =_e t'$ for all $\neg t =_e t' \in U(\Delta)$ and $A \models_{\mathcal{DNS}} \vee U(\Delta)$, as A is a model of sp, $A \models_{\mathcal{DNS}} \vee U(\Delta) \cap Eq(\Sigma, X)$ so that, because of 5, there exists $\delta \in U(\Delta) \cap Eq(\Sigma, X)$ s.t. $A \models_{\mathcal{DNS}} \delta$ for all $A \in$ NSMod(sp) and hence 6b holds.

$6 \Rightarrow 7$ It is easy to show that under the hypothesis 6a $\equiv$ is a congruence; thus it is sufficient to show that $T_\Sigma(X)/\equiv$ is the initial model.

Let I denote $T_\Sigma(X)/\equiv$, $\alpha = \vee\Phi$ be an axiom of sp, V be a valuation for $Var(\alpha)$ in I; if there exists $\neg t =_e t' \in \Phi$ s.t. $I \not\models_{\mathcal{DNS}_V} t =_e t'$, then $I \models_{\mathcal{DNS}_V} \neg t =_e t'$ and hence $I \models_{\mathcal{DNS}_V} \vee \Phi$. Thus assume that $I \models_{\mathcal{DNS}_V} t =_e t'$

for all $\neg(t =_e t') \in \Phi$; let U be a substitution for $T_\Sigma(Var(\alpha))$ in $T_\Sigma(X)$ s.t. $V(y) = [U(y)]$ for all $y \in Var(\alpha)$ and $U(\phi)$ denote $\phi[U(y)/y \mid y \in Var(\alpha)]$ for all formulas ϕ.

Because of Prop. 3.2.19, $I \models_{\mathcal{DNS}_V} t =_e t'$ implies $U(t) \equiv U(t')$ and hence, by definition of $\equiv$, $A \models_{\mathcal{DNS}} U(t =_e t')$ for all $A \in \text{NSMod}(sp)$ and all $\neg t =_e t' \in \Phi$.

Therefore, because of 6.6b, there exists $t =_e t' \in \Phi \cap Eq(\Sigma, Var(\alpha))$ s.t. $A \models_{\mathcal{DNS}} U(t =_e t')$ for all $A \in \text{NSMod}(sp)$ and hence $U(t) \equiv U(t')$, so that $I \models_{\mathcal{DNS}_V} t =_e t'$.

$7 \Rightarrow 8$ Since I satisfies conditions 1 and 2 of Theorem 3.2.36 by definition, it is initial in NSMod(sp).

$8 \Rightarrow 1$ Obvious. □

The results of the above theorem apply to a wide range of specifications, because of the great generality of non-strict disjunctive specifications.

In particular if all the proper axioms are conditional, then condition 6b is always satisfied, while if axioms are imposed so that all models are strict, then condition 6a is satisfied. Thus for total and partial conditional specifications both 6a and 6b hold, so that the known results about the existence of an initial model in those cases are a specialization of Theorem 3.2.39.

The conditional specifications are a particular case of disjunctive specifications; indeed the conditional formula $\wedge\Delta \supset \epsilon$ is logically equivalent to $\vee\{\neg\delta \mid \delta \in \Delta\} \cup \{\epsilon\}$; in other words conditional formulas are disjunctions where exactly one non-negated equality appear, i.e. are positive Horn clauses. In this case the necessary and sufficient conditions for the existence of an initial object are partially simplified.

Theorem 3.2.40 Let $sp = (\Sigma, Ax)$ be a consistent conditional specification. The following conditions are equivalent:

1. There exists I initial in NSMod(sp).
2. There exists I initial in Ind(NSMod(sp)).
3. $I(\text{NSMod}(sp)) \in \text{NSMod}(sp)$.
4. $I(\text{NSMod}(sp))$ is initial in NSMod(sp).
5. for all sets Δ of equalities

$$A \models_{\mathcal{DNS}} \vee\,\Delta \text{ for all } A \in \text{NSMod}(sp)$$

implies

$$\text{there exists } \delta \in \Delta s.t.\ A \models_{\mathcal{DNS}} \delta \text{ for all } A \in \mathrm{NSMod}(sp)$$

6. for all $f \in F_{s_1 \ldots s_n, s}$

$$A \models_{\mathcal{DNS}} D(f(t_1, \ldots, t_n)) \text{ for all } A \in \mathrm{NSMod}(sp)$$

implies that (at least) one between the following properties holds

- $A \models_{\mathcal{DNS}} D(t_i)$ for all $A \in \mathrm{NSMod}(sp)$
- $A \models_{\mathcal{DNS}} f(t_1, \ldots, t_n) =_e f(t_1, \ldots, t_{i-1}, x, t_{i+1}, \ldots, t_n)$ for all $A \in \mathrm{NSMod}(sp)$

7. the relation $\equiv$ over $T_\Sigma(X)$, defined by $t \equiv t'$ iff $A \models_{\mathcal{DNS}} t =_e t'$ for all $A \in \mathrm{NSMod}(sp)$, is a congruence and $T_\Sigma(X)/\equiv$ is a model.

8. the relation $\equiv$ over $T_\Sigma(X)$, defined by $t \equiv t'$ iff $A \models_{\mathcal{DNS}} t =_e t'$ for all $A \in \mathrm{NSMod}(sp)$, is a congruence and $T_\Sigma(X)/\equiv$ is the initial model.

Proof. First any conditional specification is shown to be equivalent to a disjunctive one which condition 6b is always satisfied for. Let sp be the consistent conditional specification (Σ, Ax) and define $Ax' = \{disj(\alpha) \mid \alpha \in Ax\}$, where $disj(\wedge\Delta \supset \epsilon) = \vee\{\neg\delta \mid \delta \in \Delta\} \cup \{\epsilon\}$, and $sp' = (\Sigma, Ax')$. Since, by definition of validity, any algebra A satisfies ϕ iff satisfies $disj(\phi)$ for all conditional formulas ϕ, $\mathrm{NSMod}(sp) = \mathrm{NSMod}(sp')$.

By Theorem 3.2.39 it is sufficient to show that for such a sp' condition 3.2.40(6) is equivalent to conditions 3.2.39 (6a and 6b), i.e. that condition 3.2.39(6b) is satisfied too. Let $\wedge\Delta \supset \epsilon$ be an axiom of sp, $U\colon Var(\wedge\Delta \supset \epsilon) \to X$ be a substitution and assume that $A \models_{\mathcal{DNS}} t =_e t'$ for all $\neg t =_e t' \in U(disj(\wedge\Delta \supset \epsilon))$, i.e. for all $t =_e t' \in U(\Delta)$, for all $A \in \mathrm{NSMod}(sp)$. Then for all $A \in \mathrm{NSMod}(sp)$, since $A \models_{\mathcal{DNS}} U(\wedge\Delta \supset \epsilon)$ and $A \models_{\mathcal{DNS}} t =_e t'$ for all $t =_e t' \in U(\Delta)$, $A \models_{\mathcal{DNS}} U(\epsilon)$, so that 3.2.39(6b) is satisfied. □

3.2.3 Relating total and non-strict algebras

This section is devote to relating the non-strict frame with the more usual total one, following the same scheme applied to the relationship between partial and total many-sorted algebras.

Def. 3.2.41 The *institution of non-strict algebras* without sentences is the quadruple $\mathcal{NS} = (\mathbf{Sign}_{\mathcal{NS}}, \emptyset, Mod_{\mathcal{NS}}, \emptyset)$, where

- $\mathbf{Sign}_{\mathcal{NS}} = \mathbf{Sign}_{\mathcal{MS}}$ is the category of many-sorted signatures.
- $Mod_{\mathcal{NS}}: \mathbf{Sign}_{\mathcal{NS}} \to \mathbf{Cat}^{Op}$ is the functor which yields for any signature Σ the category $\mathbf{NSAlg}(\Sigma)$ of non-strict algebras (see Def. 3.2.13) and for any signature morphism $(\sigma, \phi) \in \mathbf{Sign}_{\mathcal{NS}}(\Sigma_1, \Sigma_2)$ the *reduct* functor $Mod_{\mathcal{NS}}(\sigma, \phi): \mathbf{NSAlg}(\Sigma_2) \to \mathbf{NSAlg}(\Sigma_1)$, defined by:

$$Mod_{\mathcal{NS}}(\sigma, \phi)(A_2) = (\{\sigma(s)^{A_2}\}_{s \in S}, \{\phi(f)^{A_2}\}_{f \in F})$$

and

$$Mod_{\mathcal{NS}}(\sigma, \phi)(h_2) = \{h_{2\sigma(s)}\}_{s \in S}. \ \Box$$

Following the intuition that a simulation codes a new into an old frame, a simulation of non-strict by total algebras is defined.

Since the partial product of ${s_1}^A, \ldots, {s_n}^A$ is isomorphic, from a set-theoretical point of view, to the (usual) product of ${s_1}^A \cup \{\perp_{s_1}\}, \ldots, {s_n}^A \cup \{\perp_{s_n}\}$, where the symbol $\cup$ denotes the disjoint union, any non-strict algebra A is in some sense equivalent to the total algebra $A_\perp$, defined by:

Algebra $A_\perp =$

$s^{A_\perp} = s^A \cup \{\perp_s\}$

for any $a_i \in {s_i}^{A_\perp}$ for $i = 1 \ldots n$ let $\underline{a}$ be defined by $\underline{a}(i) = a_i$ if $a_i \in {s_i}^A$ and $\underline{a}(i)$ is undefined if $a_i = \perp_{s_i}$

$f^{A_\perp}(a_1, \ldots, a_n) = f^A(\underline{a})$if $f^A(\underline{a})$ is defined, else $f^{A_\perp}(a_1, \ldots, a_n) = \perp_s$

for all $f: s_1 \times \ldots \times s_n \to s$.

However this equivalence disregards the homomorphisms; indeed some homomorphism h between the trivial totalizations cannot be translated into the non-strict frame, because h maps "defined" into "undefined" elements, i.e. $h(a) = \perp_s$ for some $a \neq \perp_s$, while the non-strict homomorphisms are total functions. Moreover some non strict homomorphisms have no total correspondent, because the introduction of *one* element to represent *all* the undefined terms may cause the lack of the existence of homomorphisms, as it is shown by the following example.

Example 3.2.42 Let Σ be the one-sorted signature consisting of just three constant symbols a, b, c and A, B be the non-strict algebras over Σ, defined by:

Algebra $A =$

$s^A = \{1\}$

$a^A = 1$

b^A, c^A are undefined

Algebra B =

$s^B = \{1\}$
$a^B = 1$
$b^B = 1$
c^B is undefined

then there is a non-strict homomorphism $h: A \to B$, defined by $h(1) = 1$. Consider now the trivial totalizations of A and B.

Algebra $A_\perp$ =

$s^{A_\perp} = \{1, \perp\}$
$a^{A_\perp} = 1$
$b^{A_\perp} = \perp$
$c^{A_\perp} = \perp$

Algebra $B_\perp$ =

$s^{B_\perp} = \{1, \perp\}$
$a^{B_\perp} = 1$
$b^{B_\perp} = 1$
$c^{B_\perp} = \perp$

then there does not exist any total homomorphism from $A_\perp$ into $B_\perp$, because $b^{A_\perp} = c^{A_\perp}$, while $b^{B_\perp} \neq c^{B_\perp}$.

Summarizing the above discussion, a simulation of non-strict by total algebras is defined, that is a rigorous formalization of the usual totalization by $\perp$.

Def. 3.2.43 The simulation $\mu_0^\perp: \mathcal{NS} \to \mathcal{MS}_0$ is defined by:

- $\mu_0^\perp(S, F) = (S, F \cup \{\perp_s\}_{s \in S})$ and $\mu_0^\perp(\sigma, \phi) = (\sigma, \phi')$, where $\phi'(f) = \phi(f)$ for all $f \in F$ and $\phi'(\perp_s) = \perp_{\sigma(s)}$;
- $dom(\mu_0^\perp)$ is the category whose objects are the total algebras A' where the interpretation of function symbols are *regular* function, i.e.

$$f^{A'}(a_1, \ldots, a_{i-1}, \perp_{s_i}, a_{i+1}, \ldots, a_n) \neq \perp_s^{A'}$$

implies

$$f^{A'}(a_1, \ldots, a_{i-1}, \perp_{s_i}, a_{i+1}, \ldots, a_n) = f^{A'}(a_1, \ldots, a_{i-1}, a_i, a_{i+1}, \ldots, a_n)$$

for any a_i, and whose morphisms h *preserve definedness*, i.e. $a \neq \perp_s^{A'}$ implies $h(a) \neq \perp_s^{B'}$;

- for any A' in the objects of $dom(\mu_0^\perp)$ the translation $A = \mu_0^\perp(A')$ is the non-strict algebra, which consists of $s^A = s^{A'} - \{\perp_s^{A'}\}$ for any $s \in S$ and for any $f \in$ the function f^A is defined by:
  for any $\underline{a} \in {s_1}^A \times_p \ldots \times_p {s_n}^A$ let a_i be $\underline{a}(i)$, if $\underline{a}(i)$ is defined, $\perp_{s_i}$ otherwise; if $f^{A'}(a_1, \ldots, a_n) \neq \perp_{s_i}$, then $f^A(\underline{a}) = f^{A'}(a_1, \ldots, a_n)$, else $f^A(\underline{a})$ is undefined;
  for any arrow h' in $dom(\mu_0^\perp)$ the translation $h = \mu_0^\perp(h')$ is the restriction of h' to $\mu_0^\perp(A')$. □

It is easy to check that the components of $\mu_0^\perp$ w.r.t. the models are partially natural and hence that $\mu_0^\perp$ is a simulation.

Since $\mu_0^\perp$ does not take in account the categorical structure, the initiality in the total and in the non-strict frames are unrelated; indeed the trivial totalization of an initial model satisfies a lot of equalities between "undefined" terms, which are not satisfied by other models in the class, so that the no-confusion condition in the total frame is not satisfied and hence the trivial totalization of an initial model is in general not initial. Vice versa if the trivial totalization of a non-strict algebra is initial, then the algebra is *maximally* defined and hence it is not initial in the non-strict frame.

In order to have a representation of the category of non-strict algebras, a definition of (total) homomorphism is needed, which does not involve the "undefined" part. To do this it is useful, not to say necessary, having a tool to individuate the "undefined" elements, for example a family of unary predicates, one for each sort, dividing the carriers in "defined" and "undefined". Following a similar idea both [24] and [74] define homomorphisms which are partial functions, having as domain the "defined" part; this approach can be generalized in order to include non-strictness.

Following the same pattern seen for the partial case, the basic idea of the following simulation of non-strict by first-order structures is to split the carriers of a first-order structure into defined and undefined elements, provided that at least one undefined element, denoted by $\perp$, exists, by means of unary *definedness* predicates. Thus the simulation is defined on any first-order structure satisfying the monotonicity condition and where $\perp$ is undefined; it yields the non-strict algebra where the undefined part of the carriers has been dropped. Since the homomorphisms in the first-order frame preserve the truth of predicates, any homomorphism between such two first-order structures can be translated into a non-strict homomorphism, too. Thus the domain of this simulation is a full subcategory.

Def. 3.2.44 The simulation ${\mu^P}_0 \colon \mathcal{NS} \to \mathcal{TL}_0$ is defined by:

- ${\mu^P}_0(S,F) = (S, F \cup \{\bot_s\}_{s\in S}, \{D_s : s, eq_s : s \times s\}_{s\in S})$ and ${\mu^P}_0(\sigma,\phi) = (\sigma,\phi',\pi)$, where $\phi'(f) = \phi(f)$ for all $f \in F$, $\phi'(\bot_s) = \bot_{\sigma(s)}$, $\pi(D_s) = D_{\sigma(s)}$ and $\pi(eq_s) = eq_{\sigma(s)}$;
- $dom({\mu^P}_0)$ is the full sub-category whose objects are the first-order structures A' s.t.

  1. ${D_s}^{A'}(f^{A'}(a_1,\ldots,a_n))$ implies ${D_{s_i}}^{A'}(a_i)$ or for all a
  $$f^{A'}(a_1,\ldots,a_{i-1},a,a_{i+1},\ldots,a_n) = f^{A'}(a_1,\ldots,a_n);$$
  2. $eq_s^{A'}(a,a')$ iff ${D_s}^{A'}(a)$, ${D_s}^{A'}(a')$ and $a = a'$;
  3. $\neg {D_s}^{A'}(\bot_s^{A'})$;

  for any A' in the objects of $dom({\mu^P}_0)$ the translation $A = {\mu^P}_0(A')$ is the non-strict algebra, which consists of $s^A = {D_s}^{A'}$ for any $s \in S$ and for any $f \in F$ the function f^A is defined by:

  for any $\underline{a} \in {s_1}^A \times_p \ldots \times_p {s_n}^A$ let a_i be $\underline{a}(i)$, if $\underline{a}(i)$ is defined, $\bot_{s_i}$ otherwise; if ${D_s}^{A'}(f^{A'}(a_1,\ldots,a_n))$, then $f^A(\underline{a}) = f^{A'}(a_1,\ldots,a_n)$, else $f^A(\underline{a})$ is undefined;

  for any arrow h' in $dom({\mu^P}_0)$ the translation $h = {\mu^P}_0(h')$ is the restriction of h' to ${\mu^P}_0(A')$. □

Prop. 3.2.45 The simulation ${\mu^P}_0 : \mathcal{NS} \to \mathcal{TL}_0$ defined in Def. 3.2.44 is categorical. Moreover for any class $C' \subseteq dom({\mu^P}_0)$ of first-order structures closed w.r.t. subalgebras, if I' is initial in C', then ${\mu^P}_0(I')$ is initial in ${\mu^P}_0(C')$.

Proof. By definition $dom({\mu^P}_0)$ is a full subcategory; moreover in both frames inductive objects coincide with term-generated algebras and it is easy to check that term-generated first-order structures are translated via ${\mu^P}_0$ into term-generated non-strict algebras. Therefore ${\mu^P}_0$ is categorical and hence Prop. 3.2.7 applies, because the category of first-order structures has equalizers, which coincide with subalgebras. □

The relationship between non-strict algebras and first-order structures described by the categorical simulation ${\mu^P}_0$ is strengthened by the existence of left adjoints, from now on denoted by *Tot*, of the model components of ${\mu^P}_0$ corresponding, as usual, to free constructions. Indeed *Tot* preserves initiality, because left adjoints do; moreover, because of Prop. 3.2.45, ${\mu^P}_0$ preserves initiality, too, and hence the existence of the initial model in the non-strict and in the total frame are completely equivalent.

To build such *Tot* some preliminary technical results are needed.

Lemma 3.2.46 Let $\Sigma = (S, F)$ be a non-strict signature, A be a non-strict algebra over Σ, X^A be the S-sorted family defined by $X_s = s^A \cup \{\perp\}$ for all $s \in S$ and $V_A\colon X^A \to A$ be the valuation defined by $V_A(a) = a$ if $a \in s^A$, $V_A(\perp)$ undefined.

Let $\equiv^A$ denote the total many-sorted congruence over $T_\Sigma(X^A)$ generated by:

$$\{(t, t') \mid t \in T_\Sigma(X^A), A \models_{\mathcal{DNS}_{V_A}} t =_e t'\}$$

and $Tot(A)$ denote the total first-order structure

$$(T_\Sigma(X^A)/\equiv^A, \{\perp_s^{Tot(A)}\}_{s\in S}, \{D_s{}^{Tot(A)}, eq_s{}^{Tot(A)}\}_{s\in S}),$$

where $\perp_s{}^{Tot(A)} = [\perp]_{\equiv^A}$, $D_s{}^{Tot(A)}([t])$ iff $A \models_{\mathcal{DNS}_{V_A}} D(t)$ and $eq_s{}^{Tot(A)}([t]_{\equiv^A}, [t']_{\equiv^A})$ iff $A \models_{\mathcal{DNS}_{V_A}} t =_e t'$.

The following facts hold

1. $t \equiv^A t'$ and $A \models_{\mathcal{DNS}_{V_A}} D(t)$ or $A \models_{\mathcal{DNS}_{V_A}} D(t')$ imply $A \models_{\mathcal{DNS}_{V_A}} t =_e t'$;

2. $Tot(A)$ belongs to $dom(\mu^P{}_0)$;

3. $\eta^A\colon A \to \mu^P{}_0(Tot(A))$, defined by $\eta^A(a) = [a]_{\equiv^A}$, is an isomorphism.

Proof.

1. It is immediate to verify, by induction over the definition of $\equiv^A$, that $t \equiv^A t'$ implies $t^{A,V_A} = t'^{A,V_A}$. Thus the thesis follows.

2. Because of 1 and of the definition of both $D_s{}^{Tot(A)}$ and $eq_s{}^{Tot(A)}$, the conditions 2 and 3 of Def. 3.2.44 are satisfied. Assume that $[f(t_1, \ldots, t_n)]_{\equiv^A} \in D_s{}^{Tot(A)}$ and $[t_i]_{\equiv^A} \notin D_{s_i}{}^{Tot(A)}$; then, by definition of $D_s{}^{Tot(A)}$, $f(t_1, \ldots, t_n)^{A,V_A} \in s^A$ and $t_i{}^{A,V_A} \notin s_i{}^A$, i.e.

$$f^A(t_1{}^{A,V_A}, \ldots, t_{i-1}^{A,V_A}, ?, t_{i+1}^{A,V_A}, \ldots, t_n{}^{A,V_A}) \in s^A$$

and hence, because of the monotonicity of f^A,

$$f^A(t_1{}^{A,V_A}, \ldots, t_{i-1}^{A,V_A}, ?, t_{i+1}^{A,V_A}, \ldots, t_n{}^{A,V_A})$$

is existentially equal to

$$f^A(t_1{}^{A,V_A}, \ldots, t_{i-1}^{A,V_A}, t^{A,V_A}, t_{i+1}^{A,V_A}, \ldots, t_n{}^{A,V_A})$$

for all $t \in T_{\mu^P{}_0(\Sigma)}(X^A)$ so that $f(t_1, \ldots, t_n) \equiv^A f(t_1, \ldots, t_{i-1}, t, t_{i+1}, \ldots, t_n)$ and hence also condition 1 is satisfied.

3. Obvious because of 1. $\square$

Consider a non-strict homomorphism $h: A \to B$. In order to define its image along *Tot*, h is used as a valuation from X^A into X^B and then it is shown that $t \equiv^A t'$ implies $h(t) \equiv^B h(t')$, so that $Tot(h)([t]_{\equiv^A}) = [h(t)]_{\equiv^B}$ is well defined.

$$\begin{tikzcd} A \arrow[d, "h"] & X^A \arrow[r] \arrow[d, "h"'] & T_\Sigma(X^A)/\equiv^A \arrow[d, "Tot(h)"] \\ B & X^B \arrow[r] & T_\Sigma(X^B)/\equiv^B \end{tikzcd}$$

Lemma 3.2.47 Let A and B be non-strict algebras over Σ and $h: A \to B$ be a non-strict homomorphism. Using the notation of Lemma 3.2.46

1. for any $t \in T_\Sigma(X^A)$ let $h(t)$ denote the term $t[h(a)/a \mid a \in s^A] \in T_\Sigma(X^B)$; then for all $t, t' \in T_\Sigma(X^A)$ if $t \equiv^A t'$, then $h(t) \equiv^B h(t')$;

2. $Tot(h): Tot(A) \to Tot(B)$, defined by $Tot(h)([t]_{\equiv^A}) = [h(t)]_{\equiv^B}$, is a homomorphism of first-order structures.

Proof.

1. It is easy to check that, by definition of congruence,

   $$\{(h(t), h(t')) \mid t \equiv^A t'\} \subseteq \approx,$$

   where $\approx$ is the congruence generated by

   $$\{(h(t), h(t')) \mid A \models_{\mathcal{DNS}_{V_A}} t =_e t'\},$$

   because $\equiv^A$ is generated by $\{(t, t') \mid A \models_{\mathcal{DNS}_{V_A}} t =_e t'\}$. Thus it is sufficient to show that $\approx \subseteq \equiv^B$. To do this assume that $A \models_{\mathcal{DNS}_{V_A}} t =_e t'$ for some $t, t' \in T_\Sigma(X^A)$ and show that $B \models_{\mathcal{DNS}_{V_B}} h(t) =_e h(t')$. Because of Prop. 3.2.17, $B \models_{\mathcal{DNS}_{h \circ V_A}} t =_e t'$, and hence, since $t^{B, h \circ V_A} = h(t)^{B, V_B}$ by definition of V_A and V_B, $B \models_{\mathcal{DNS}_{V_B}} h(t) =_e h(t')$.

2. Because of 1, $Tot(h)$ is a well-defined many-sorted homomorphism; thus it is sufficient to show that it preserves the operations $\perp_s$ and the truth of the predicates. By definition $Tot(h)([\perp]_{\equiv^A}) = [h(\perp)]_{\equiv^B} = [\perp]_{\equiv^B}$. Because of Lemma 3.2.46, $[t]_{\equiv^A} \in {D_s}^{Tot(A)}$ implies $A \models_{\mathcal{DNS}_{V_A}} D(t)$ and hence, because of Prop. 3.2.17, $B \models_{\mathcal{DNS}_{h \circ V_A}} D(t)$, i.e. $B \models_{\mathcal{DNS}_{V_B}} D(h(t))$, so that

$Tot(h)([t]_{\equiv^A}) = [h(t)]_{\equiv^B} \in D_s{}^{Tot(B)}$. Analogously $([t]_{\equiv^A}, [t']_{\equiv^A}) {=_e}^{Tot(A)}$ implies $A \models_{\mathcal{DNS}_{V_A}} t =_e t'$, so that $B \models_{\mathcal{DNS}_{V_B}} h(t) =_e h(t')$, and hence $([h(t)]_{\equiv^B}, [h(t')]_{\equiv^B}) {=_e}^{Tot(B)}$. Therefore $Tot(h)$ is a homomorphism of first-order structures. □

And finally, putting together Lemma 3.2.46 and Lemma 3.2.47, the functor *Tot* is defined.

Theorem 3.2.48 Let $\Sigma = (S, F)$ be a non-strict signature; using the notation of Lemma 3.2.46 and Lemma 3.2.47, *Tot* is a functor and is the left adjoint and left inverse of

$\mu^P{}_0$. Moreover if I is initial in a class C of non-strict algebras closed w.r.t. isomorphisms, then $Tot(I)$ is initial in ${\mu^P{}_0}^{-1}(C)$.

Proof. It is just a trivial check to prove that *Tot* is a functor.

Thus it is sufficient to show that *Tot* is the left adjoint of $\mu^P{}_0$ and that the family of the isomorphisms η^A, defined in Lemma 3.2.46, is the unit of the adjunction. Let A be a non-strict algebra, B' a first-order structure belonging to $dom(\mu^P{}_0)$ and $h\colon A \to \mu^P{}_0(B')$ a non-strict homomorphism. The homomorphism defined by $k^{B'}([t]_{\equiv^{B'}}) = t^{V,B'}$ for $V\colon \mu^P{}_0(B') \to B'$ the identical valuation is denoted by $k^{B'}\colon Tot(\mu^P{}_0(B')) \to B'$; it is sufficient to show that $h^\sharp = k^{B'} \circ Tot(h)$ is the unique homomorphism from $Tot(A)$ into B' s.t. the following diagram commutes.

$$\begin{tikzcd}
A \arrow[r, "\eta^A"] \arrow[dr, "h"'] & \mu^P{}_0(Tot(A)) \arrow[d, "\mu^P{}_0(h^\sharp)"] & Tot(A) \arrow[d, "h^\sharp"] \\
& \mu^P{}_0(B') & B'
\end{tikzcd}$$

By definition of η^A and $h^\sharp$,

$$\mu^P{}_0(h^\sharp) \circ \eta^A(a) = \mu^P{}_0(h^\sharp)([a]_{\equiv^A}) = \mu^P{}_0(k^{B'} \circ Tot(h))([a]_{\equiv^A})$$

and

$$\mu^P{}_0(k^{B'} \circ Tot(h))([a]_{\equiv^A}) = k^{B'} \circ Tot(h)([a]_{\equiv^A}) = k^{B'}([h(a)]_{\equiv^{B'}}),$$

because $\mu^P{}_0$ is the restriction; finally $k^{B'}([h(a)]_{\equiv^{B'}}) = h(a)$, by definition of $k^{B'}$, so that the diagram commutes.

Moreover $h^\sharp$ is the unique arrow which makes the diagram commute. Indeed let k be s.t. $\mu^P{}_0(k) \circ \eta^A = h$; then, by definition of $\mu^P{}_0$ and η^A, $\mu^P{}_0(k) \circ \eta^A =$

$k([a]_{\equiv^A}) = h(a)$ for any $a \in s^A$ and hence k and $h^\sharp$ coincide on the (equivalence classes of) variables and hence, by induction, on $Tot(A)$.

Finally, since left adjoints preserve initiality and Tot is the left adjoint of ${\mu^P}_0$, if I is initial in C, then $Tot(I)$ is initial in any class C' s.t. both $Tot: C \to C'$ and ${\mu^P}_0: C' \to C$; in particular, if C is closed w.r.t. isomorphisms, also ${\mu^P}_0^{-1}(C)$ is closed w.r.t. isomorphisms and hence, as A is isomorphic to ${\mu^P}_0(Tot(A))$ by Lemma 3.2.46, $Tot: C \to {\mu^P}_0^{-1}(C)$ and obviously ${\mu^P}_0: {\mu^P}_0^{-1}(C) \to C$, so that $Tot(I)$ is initial in ${\mu^P}_0^{-1}(C)$. □

Coming to the logical aspects of the relationships between non-strict and total algebras, the institutions $\mathcal{CNS}$ and $\mathcal{DNS}$ of non-strict algebras, respectively with conditional and disjunctive axioms as sentences and the institutions $\mathcal{TL}$ and $\mathcal{DTL}$ of first-order structures, respectively with conditional and disjunctive axioms built on atomic formulas of the form $p(t_1, \ldots, t_k)$ as sentences, are considered.

Consider first the trivial totalization $\mu_0^\perp$. Let A' belong to $dom(\mu_0^\perp)$ and consider a ground existential equality $t =_e t'$; then $A = \mu_0^\perp(A')$ satisfies $t =_e t'$ iff both t and t' denote the same element of $s^A = s^{A'} - \{\perp_s^{A'}\}$, i.e. iff $t^{A'} = t'^{A'} \neq \perp_s^{A'}$. Thus to extend $\mu_0^\perp$ to a simulation working on equations of the non-strict frame, inequalities are needed in the total frame. This is another inadequacy of the trivial totalization, which has already been proved unable to deal with the categorical structure of the non-strict frame.

Consider now the simulation ${\mu^P}_0$, defined in Def. 3.2.44; it is easy to extend ${\mu^P}_0$ to work on conditional (disjunctive) formulas, i.e. to define two simulations $\mu_C: \mathcal{CNS} \to \mathcal{TL}$ and $\mu_D: \mathcal{DNS} \to \mathcal{DTL}$ coinciding with ${\mu^P}_0$ on signatures and models. Indeed any conditional (disjunctive) formula can be naturally

translated from the non-strict into the first-order frame, by just replacing the existential equalities by the eq_s predicates, which were indeed introduced to represent existential equalities.

Def. 3.2.49 The simulation $\mu_C: \mathcal{CNS} \to \mathcal{TL}$ is the extension of ${\mu^P}_0: \mathcal{NS} \to \mathcal{TL}$, which on $\wedge\{t_i =_e t'_i \mid i \in I\} \supset t =_e t'$ yields

$$\wedge\{eq_{s_i}(t_i, t'_i) \mid i \in I\} \supset eq_s(t, t').$$

The simulation $\mu_D: \mathcal{DNS} \to \mathcal{DTL}$ is the extension of ${\mu^P}_0: \mathcal{NS} \to \mathcal{TL}$, which on $\vee\{t_i =_e t'_i \mid i \in I\} \vee \{\neg t_j =_e t'_j \mid j \in J\}$ yields

$$\vee\{eq_{s_i}(t_i, t'_i) \mid i \in I\} \vee \{\neg eq_{s_j}(t_j, t'_j) \mid j \in J\}. \ \square$$

Since the domain of μ_D is the model class of the set Ax_{DC} consisting of the following disjunctive axioms:

- $\neg D_s(f(x_1,\ldots,x_n)) \vee D_{s_i}(x_i) \vee eq_s(f(x_1,\ldots,x_{i-1},y,x_{i+1},\ldots,x_n), f(x_1,\ldots,x_n))$ for every $f \in F_{s_1\ldots s_n,s}$;
- $eq_s(x,x') \vee \neg D_s(x) \vee \neg D_s(x') \vee \neg x = x'$ for every $s \in S$;
- $\neg eq_s(x,x') \vee D_s(x)$ for every $s \in S$;
- $\neg eq_s(x,x') \vee D_s(x')$ for every $s \in S$;
- $\neg eq_s(x,x') \vee x = x'$ for every $s \in S$;
- $\neg D_s(\perp_s)$ for every $s \in S$;

the non-strict model class of a set Ax of disjunctive formulas is simulated by the total model class of the set $\mu_D(Ax) \cup Ax_{DC}$ of disjunctive formulas in the first-order frame. Therefore μ_D induces a correspondence between the specifications in the two formalisms and hence both frames have the same expressive power.

Instead the simulation μ_C does not relate conditional to conditional specifications, because proper disjunctive specifications are required to describe the domain of the simulation; indeed there does not exist a total conditional specification whose model class is $dom({\mu^P}_0)$, because the trivial total algebra Tr over ${\mu^P}_0(\Sigma)$, having singleton sets as carriers, the unique obvious interpretation of function symbols and the totally true predicates, i.e. ${D_s}^{Tr} = s^{Tr}$ and ${eq_s}^{Tr} = s^{Tr} \times s^{Tr}$, is a model of each conditional specification while

does not belong to the domain. Therefore in general it is impossible to translate a conditional (equational) non-strict specification into a conditional first-order one, because it is impossible at least for the specification without axioms $(\Sigma, \emptyset)$.

Since the first-order structures representing the models of a non-strict disjunctive specifications are the models of a (total) disjunctive specification and initiality is both preserved,

because ${\mu^P}_0$ is categorical, and reflected, because of the existence of the left adjoint Tot, by ${\mu^P}_0$, the existence of an initial model for a non-strict disjunctive specification is equivalent to the existence of an initial model for the first-order disjunctive specification which is simulating it.

Cor. 3.2.50 Let $sp = (\Sigma, Ax)$ be a non-strict disjunctive specification and ${\mu^P}_0(sp)$ denote the first-order disjunctive specification

$$({\mu^P}_0(\Sigma), {\mu^P}_0(Ax) \cup Ax_{DC}).$$

Then I is initial for sp iff $Tot(I)$ is initial for ${\mu^P}_0(sp)$.

Proof. Since the model classes of disjunctive specifications in both frames are closed w.r.t. isomorphisms and regular subobjects, Prop. 3.2.45 and Theorem 3.2.48 apply.

If I is initial for sp, then $Tot(I)$ is initial for ${\mu^P}_0(sp)$, because of Theorem 3.2.48, and if $Tot(I)$ is initial for ${\mu^P}_0(sp)$, then $I \approx {\mu^P}_0(Tot(I))$ is initial for sp, because of Prop. 3.2.45. □

3.3 Arrows between Institutions

Due to the relevance of the interaction of different formal systems, several concepts of arrows between institutions have been developed or are under development. However the nature of the relationships induced by the various arrows is quite different; moreover some of them are not conveying the meaning of translation and some others are serving different purposes than simulations.

3.3.1 Maps of Institutions

The *maps of institutions*, developed by Meseguer in the big fresco of General Logic (see [63]), is the closest concept to simulations; indeed the components of simulations and maps have the same direction, models are partially mapped in both cases and the satisfaction condition is the same. However the two notions are not exactly the same; indeed maps of institutions are not required to be surjective (but if they are, then are also simulations), on the converse the domains of the maps model components are required to be the model classes of (a natural family of) sets of new sentences and hence only logical simulations are also maps.

The definition of map of institution shares with the map of entailment systems the component dealing with signature and sentences, so that it can seem more complex than needed. In order to capture the relevant cases of translation of the logical part three levels of increasing difficulty are possible choices

- each signature Σ is translated into a signature $\Phi(\Sigma)$ and each sentence ϕ into a sentence $\alpha(\phi)$ (on $\Phi(\Sigma)$ in a uniform way); then a theory (Σ, Ax) is translated into $(\Phi(\Sigma), \{\alpha(\phi) \mid \phi \in Ax\})$. In this case the map is called *simple*.

- each signature Σ is translated into a theory $\Phi(\Sigma)$ and each sentence ϕ into a sentence $\alpha(\phi)$ (on $\Phi(\Sigma)$ in a uniform way); then a theory (Σ, Ax) is translated into $(\Sigma', \{\alpha(\phi) \mid \phi \in Ax\} \cup Ax')$, where $\Phi(\Sigma) = (\Sigma', Ax')$. In this case the map is called *plain*.

- theories are translated into theories accordingly with the translation of sentences but the image theory cannot be divided in the part representing the signature and the translation of the sentences; the simplest example of this kind of translation is the closure of theories under deduction; a perhaps more significant example is the *unfailing Knuth-Bendix completion.*

Notation. Due to the extensive use of theories in the sequel, some short notation is in order here; given an institution $\mathcal{I} = (\mathbf{Sign}, Sen, Mod, \models)$, the symbol Sen is used to denote the composition $Sen \circ sign$, where $sign$ denotes the projection of theories in $\mathcal{I}$ to their signatures, too.

Def. 3.3.1 Given institutions $\mathcal{I} = (\mathbf{Sign}, Sen, Mod, \models)$ and $\mathcal{I}' = (\mathbf{Sign}', Sen', Mod', \models')$, a *map of institution* $(\Phi, \alpha, \beta): \mathcal{I} \to \mathcal{I}'$ consists of a natural transformation $\alpha: Sen \to Sen' \circ \Phi$, an α-sensible functor $\Phi: \mathbf{Th}_0 \to \mathbf{Th'}_0$, and a natural transformation $\beta: \mathbf{Mod}' \circ \Phi \to \mathbf{Mod}$ such that for each $\Sigma \in |\mathbf{Sign}|$, each $\phi \in Sen(\Sigma)$ and each $M' \in \mathbf{Mod}'(\Phi(\Sigma, \emptyset))$ the following property is satisfied:

$$M' {\models'}_{\Sigma'} \alpha_\Sigma(\phi) \text{ iff } \beta_{(\Sigma,\emptyset)}(M') \models_\Sigma \phi$$

where Σ' is the signature of the theory $\Phi(\Sigma, \emptyset)$ and Φ is α-sensible iff

1. there is a functor $\Phi^\diamond: \mathbf{Sign} \to \mathbf{Sign}'$ s.t. $sign' \circ \Phi = \Phi^\diamond \circ sign$, where $sign$ ($sign'$) denotes the projection of theories in $\mathcal{I}$ ($\mathcal{I}'$) to their signatures;

2. for any theory $T = (\Sigma, \Gamma)$, $\Phi(T)$ has the same theorems as $\Phi(\Sigma, \emptyset) \cup (\Phi^\diamond(\Sigma), \alpha_\Sigma(\Gamma))$.

Given a map of institution $\rho = (\Phi, \alpha, \beta): \mathcal{I} \to \mathcal{I}'$, the map ρ is called *plain* iff $\Phi(\Sigma, \Gamma)$ coincides with $\Phi(\Sigma, \emptyset) \cup (\Phi^\diamond(\Sigma), \alpha_\Sigma(\Gamma))$ and is *simple* iff it is plain and $\Phi^\diamond(\Sigma)$ has no axioms. $\square$

Note that from a model theoretic point of view two theories having the same deductive closure are equivalent, in the sense that they have the same model class; thus plain maps are sufficient to deal with the translation of institutions and maps of institutions are based on the general condition of α-sensibility because are part of a big framework where tools to deal with entailment systems and proof calculi are strictly interconnected.

It is immediate to check that, by suitably lifting functors and natural transformations to work on the right objects, every surjective map of institutions is a simulation, too.

Prop. 3.3.2 Let $\mathcal{I} = (\mathbf{Sign}, Sen, Mod, \models)$, $\mathcal{I}' = (\mathbf{Sign}', Sen', Mod', \models')$ be institutions and $(\Phi, \alpha, \beta): \mathcal{I} \to \mathcal{I}'$ be a map of institution s.t. $\beta_{(\Sigma,\emptyset)}$ is surjective on the objects for each $\Sigma \in |\mathbf{Sign}|$; then $\mu: \mathcal{I} \to \mathcal{I}'$ is a simulation, where

- $\mu_{Sign}: \mathbf{Sign} \to \mathbf{Sign}'$ is the composition $sign \circ \Phi$;
- $\mu_{Sen}: Sen \to Sen' \circ \mu_{Sign}$ is α;
- $\mu_{Mod}: \mathbf{Mod}' \circ \mu_{Sign} \to Mod$ is defined by $\mu_{Mod\Sigma} = \beta_{(\Sigma,\emptyset)}$ for all $\Sigma \in |\mathbf{Sign}|$.

Proof. Trivial check from the definitions. $\square$

On the converse any logical simulation defines a plain map of institutions.

Prop. 3.3.3 Let $\mathcal{I} = (\mathbf{Sign}, Sen, Mod, \models)$, $\mathcal{I}' = (\mathbf{Sign}', Sen', Mod', \models')$ be institutions and $\mu: \mathcal{I} \to \mathcal{I}'$ be a logical simulation.

Then $(\Phi, \alpha, \beta): \mathcal{I} \to \mathcal{I}'$ is a plain map of institution, where

- $\Phi^\diamond: \mathbf{Sign} \to \mathbf{Th}'_0$ is defined by $\Phi^\diamond(\Sigma) = (\mu_{Sign}(\Sigma), \Gamma_\Sigma)$ for all $\Sigma \in |\mathbf{Sign}|$, where

  $$\Gamma_\Sigma = \{\phi' \mid \phi' \in Sen'(\mu_{Sign}(\Sigma)), A \models_{\mu_{Sign}(\Sigma)} \phi' \text{ for all } A \in dom(\mu)_\Sigma\},$$

  and $\Phi^\diamond(\sigma) = \mu_{Sign}(\sigma)$ for all signature morphism σ in $\mathbf{Sign}$;
- $\alpha: Sen \to Sen' \circ \Phi^\diamond$ is μ_{Sen} and
  $\Phi(\Sigma, \Delta) = (sign(\Phi^\diamond(\Sigma)), axiom(\Phi^\diamond(\Sigma)) \cup \alpha(\Delta))$
- $\beta: \mathbf{Mod}' \circ \Phi \to Mod$ has components $\beta_{(\Sigma,\Delta)}$ that are the restrictions of μ_{Mod} to the models of Δ for all $\Sigma \in |\mathbf{Sign}|$.

Proof. Since μ is logical, for each $\Sigma \in |\mathbf{Sign}|$ there exists a subset Γ'_Σ of $Sen'(\mu_{Sign}(\Sigma))$ s.t. $dom(\mu)_\Sigma$ is the model class of the theory (Σ, Γ'_Σ) and by definition $\Gamma'_\Sigma \subseteq \Gamma_\Sigma$; thus $dom(\mu)_\Sigma$ is the model class of $\Phi^\diamond(\Sigma)$. Moreover if $A' \in dom(\mu)_{\Sigma'}$, then $Mod'(\mu_{Sign}(\sigma))(A') \in dom(\mu)_\Sigma$ for every signature morphism $\sigma: \Sigma \to \Sigma'$, by partial-naturality, and hence if $\gamma \in \Gamma_\Sigma$, i.e. $A \models_\Sigma \gamma$ for all $A \in dom(\mu)_\Sigma$, then, in particular, $Mod'(\mu_{Sign}(\sigma))(A') \models_\Sigma \gamma$ for all $A' \in dom(\mu)_{\Sigma'}$, so that $A' \models_{\Sigma'} Sen'(\mu_{Sign}(\sigma))(\gamma)$, by satisfaction condition, for all $A' \in dom(\mu)_{\Sigma'}$, i.e. $Sen'(\mu_{Sign}(\sigma))(\gamma) \in \Gamma_{\Sigma'}$. Therefore $\Phi^\diamond(\sigma) = \mu_{Sign}(\sigma)$ is a theory morphism from $\Phi^\diamond(\Sigma)$ into $\Phi^\diamond(\Sigma')$.

From this it is trivial to check that $\Phi^\diamond$ is a functor and that (Φ, α, β) is a map of institutions, that is plain by definition. $\square$

Note that the Example 3.2.11 guarantees that a map of institutions between $\mathcal{GPAR}$ (the ground partial Horn Clauses institution) and $\mathcal{GTL}$ (the ground total many-sorted Horn Clauses institution) does not exist coinciding with the simulation μ^P on models and sentences. Thus non all simulations can be made maps of institutions, because the expressive power of the target institution may be too poor to define the domain of the simulation.

3.3.2 Institutions Morphisms

The *Institution Morphisms*, introduced in [44], captures the idea of enriching an institution by new features and was designed to build new institutions where the logical tools from two or more basic institutions are available at a time. Technically morphisms differ from simulations in one essential point: they translate signatures and models together, and sentences in the opposite direction; while inverting the arrow both between models and between sentences seems essential for capturing the idea of translating formalisms.

Def. 3.3.4 Given institutions $\mathcal{I} = (\mathbf{Sign}, Sen, Mod, \models)$ and $\mathcal{I}' = (\mathbf{Sign}', Sen', Mod', \models')$, a *morphism of institution* $(\Phi, \alpha, \beta): \mathcal{I} \to \mathcal{I}'$ consists of

- a functor $\Phi: \mathbf{Sign} \to \mathbf{Sign}'$;
- a natural transformation $\alpha: Sen' \circ \Phi \to Sen$ and
- a natural transformation $\beta: Mod \to Mod' \circ \Phi$

such that for each $\Sigma \in |\mathbf{Sign}|$, each $\phi' \in Sen' \circ \Phi(\Sigma)$ and each $M \in Mod(\Sigma)$ the following property is satisfied:

$$M \models_\Sigma \alpha_\Sigma(\phi') \text{ iff } \beta_\Sigma(M) {\models'}_{\Phi(\Sigma)} \phi'.$$

Notice that simulations and institutions morphisms are not dual concepts, because in the case of the simulation models are countervariant w.r.t. sentences and signatures, while in the case of institution morphisms sentences are countervariant w.r.t. models and signatures. However these two notions are strictly related and in particular institution isomorphisms and isosimulations coincide.

Prop. 3.3.5 Let $\mathcal{I} = (\mathbf{Sign}, Sen, Mod, \models)$, $\mathcal{I}' = (\mathbf{Sign}', Sen', Mod', \models')$ be institutions.

1. If $\mu: \mathcal{I} \to \mathcal{I}'$ is a simulation s.t.

- a functor $F\colon \mathbf{Sign}' \to \mathbf{Sign}$ exists s.t. $\mu_{Sign} \circ F = Id_{\mathbf{Sign}'}$, i.e. μ_{Sign} is a *retraction*;
- $dom(\mu)_\Sigma = Mod'(\mu_{Sign}(\Sigma))$ for all $\Sigma \in |\mathbf{Sign}|$;

then $(F, \mu_{Sen_F}$[1]$, \mu_{Mod_F})$ is an institution morphism from $\mathcal{I}'$ into $\mathcal{I}$.

2. If $(\Phi, \alpha, \beta)\colon \mathcal{I} \to \mathcal{I}'$ is an institution morphism s.t.

   - a functor $F\colon \mathbf{Sign}' \to \mathbf{Sign}$ exists s.t. $\Phi \circ F = Id_{\mathbf{Sign}'}$, i.e. Φ is a *retraction*;
   - β_Σ is surjective on the objects for all $\Sigma \in |\mathbf{Sign}|$;

   then $(F, \mu_{Sen_F}, \mu_{Mod_F})$ is a simulation from $\mathcal{I}'$ into $\mathcal{I}$.

3. μ is an iso-simulation iff $(\mu_{Sign}{}^{-1}, \mu_{Sen_{\mu_{Sign}^{-1}}}, \mu_{Mod_{\mu_{Sign}^{-1}}})$ is an institution isomorphism.

Proof. It is immediate to check that 1 and 2 holds from the definitions and then 3 follows. □

3.3.3 Pre-Institution Transformations

In the frame of pre-institution (see [81]), generalizing some concrete examples in [60], the notion of *pre-institution transformation* is introduced, to relate pre-institutions and hence, in particular, institutions. A transformation translates signatures, set of sentences and models all covariantly and associate each signature with a signature, each set of sentences with a set of sentences and each model with a non-empty set of models.

The motivating example to relate set of sentences instead of sentences is the "equivalence" between the institutions of partial algebras respectively with *positive* conditional axioms, i.e. Horn-Clauses built on definedness predicates and strong equality s.t. for every strong equality $t{=}t'$ in the premises $D(t)$ or $D(t')$ is in premises too, and Horn-Clauses built on existential equality as sentences. In this case both the signature and the models are unaffected (i.e. translated by identities), but each positive conditional axiom of the form $\epsilon_1 \wedge \ldots \wedge \epsilon_n \supset t{=}t'$ logically corresponds to the couple of Horn-Clauses $\eta_1 \wedge \ldots \wedge \eta_n \wedge D(t) \supset t{=_e}t'$ and $\eta_1 \wedge \ldots \wedge \eta_n \wedge D(t') \supset t{=_e}t'$, where if ϵ_i is $D(t)$, then η_i is $t{=_e}t$ and if ϵ_i is $t{=}t'$, then η_i is $t{=_e}t'$.

[1]Here, as usual, the composition of a natural transformation $\alpha\colon F \Rightarrow G$ with a functor H (right-composable with F and G), yielding a natural transformation from $F \circ H$ into $G \circ H$, is denoted by α_H

Def. 3.3.6 Let $\mathcal{I} = (\mathbf{Sign}, Sen, Mod, \models)$, $\mathcal{I}' = (\mathbf{Sign}', Sen', Mod', \models')$ be pre-institutions and Pre (Pre') denotes the *presentation functor* that is the composition of the sentence functor with the power functor, i.e. $\mathsf{Pre} = \wp \circ Sen: \mathbf{Sign} \to \mathbf{Set}$ ($\mathsf{Pre}' = \wp \circ Sen': \mathbf{Sign}' \to \mathbf{Set}$), where $\wp: \mathbf{Set} \to \mathbf{Set}$ sends every set to the collection of its subsets and every function f to the function yielding the f-image of each subset.

A *pre-institution transformation* $\mathcal{T}: \mathcal{I} \to \mathcal{I}'$ consists of:

- a functor $\mathsf{Si}_{\mathcal{T}}: \mathbf{Sign} \to \mathbf{Sign}'$;
- a natural transformation $\mathsf{Pr}_{\mathcal{T}}: \mathsf{Pre} \Rightarrow \mathsf{Pre}' \circ \mathsf{Si}_{\mathcal{T}}$;
- a natural transformation $\mathsf{Mo}_{\mathcal{T}}: Mod \Rightarrow \wp \circ Mod' \circ \mathsf{Si}_{\mathcal{T}}$ s.t. $\mathsf{Mo}_{\mathcal{T}\Sigma}(M)$ is a non-empty set for every $\Sigma \in |\mathbf{Sign}|$ and each $M \in Mod(\Sigma)$;

s.t. the following *satisfaction invariant* holds

$$M \models_{\Sigma} E \iff \mathsf{Mo}_{\mathcal{T}}(M) {\models'}_{\mathsf{Si}_{\mathcal{T}}(\Sigma)} \mathsf{Pr}_{\mathcal{T}}(E)$$

for all $\Sigma \in |\mathbf{Sign}|$, all $E \in \mathsf{Pre}(\Sigma)$ and all $M \in Mod(\Sigma)$. □

Three main differences can be seen distinguishing pre-institution transformations between institutions and simulations:

- sets of sentences are translated by pre-institution transformations instead of single sentences;
- models are translated by pre-institution covariantly w.r.t. signature instead that countervariantly;
- satisfaction invariant relates the satisfaction by $\mathsf{Mo}_{\mathcal{T}}(M)$ as a whole with that by the individual model M.

Since any institution $\mathcal{I}$ implicitly define an institution $\wp(\mathcal{I})$ where the sentence functor has been substituted with the presentation functor, the first point may be solved relating institution transformations between $\mathcal{I}$ and $\mathcal{I}'$ to simulations between $\wp(\mathcal{I})$ and $\wp(\mathcal{I}')$. Note that the institutions $\mathcal{I}$ and $\wp(\mathcal{I})$ are logically equivalent, because any theory th in $\wp(\mathcal{I})$ corresponds to the theory of $\mathcal{I}$ whose axiom set is the union of the axioms of th, in the sense that both theories describe the same model class. Moreover each deductive tool for $\mathcal{I}$ may be easily (and automatically) lifted to work on $\wp(\mathcal{I})$, so that the equivalence of $\mathcal{I}$ and $\wp(\mathcal{I})$ is complete. The following lemma formally define $\wp(\mathcal{I})$.

Lemma 3.3.7 Let $\mathcal{I}$ be an institution and $\wp(\mathcal{I})$ denote the quadruple $(\mathbf{Sign}, \mathsf{Pre}, Mod, \models^{\wp})$, where, for all $\Sigma \in |\mathbf{Sign}|$, all $E \in \mathsf{Pre}(\Sigma)$ and all $M \in Mod(\Sigma)$, $M {\models^{\wp}}_{\Sigma} E$ iff $M \models_{\Sigma} \phi$ for all $\phi \in E$.

Then $\wp(\mathcal{I})$ is an institution with the same expressive power of $\mathcal{I}$, i.e. for every theory $th = (\Sigma, Ax)$ in $\wp(\mathcal{I})$ a theory $th' = (\Sigma, Ax')$ in $\mathcal{I}$ exists s.t. the models of th are the models of th' and vice versa.

Proof. In order to show that $\wp(\mathcal{I})$ is an institution it is sufficient to prove that the satisfaction condition holds; let $\sigma\colon \Sigma \to \Sigma'$ be a signature morphism, $E \in \mathsf{Pre}(\Sigma)$ be a sentence of $\wp(\mathcal{I})$ and $M' \in Mod(\Sigma')$ be a model. Then $Mod(\sigma)(M') {\models^{\wp}}_{\Sigma} E$ iff $Mod(\sigma)(M') \models_{\Sigma} \phi$ for all $\phi \in E$, i.e. by the satisfaction condition in $\mathcal{I}$, iff $M' \models_{\Sigma'} Sen(\sigma)(\phi)$ for all $\phi \in E$, i.e. iff $M' {\models^{\wp}}_{\Sigma'} \mathsf{Pre}(\sigma)(E)$, because $\mathsf{Pre}(\sigma)(E) = \{Sen(\sigma)(\phi) \mid \phi \in E\}$.

Let $th = (\Sigma, Ax)$ be a theory of $\wp(\mathcal{I})$ and define the theory $th' = (\Sigma, Ax')$ of $\mathcal{I}$ by $Ax' = \cup_{\alpha \in Ax} \alpha$; by definition $M \models \alpha'$ for all $\alpha' \in Ax'$ iff $M \models^{\wp} \alpha$ for all $\alpha \in Ax$.

Vice versa let $th' = (\Sigma, Ax')$ be a theory of $\mathcal{I}$ and define the theory $th = (\Sigma, Ax)$ of $\wp(\mathcal{I})$ by $Ax = \{\{\alpha\} \mid \alpha \in Ax\}$; by definition $M \models \alpha'$ for all $\alpha' \in Ax'$ iff $M \models^{\wp} \alpha$ for all $\alpha \in Ax$. $\square$

In the sequel the validity $\models^{\wp}$ in the $\wp(\mathcal{I})$ institution will be denoted by $\models$, provided that no ambiguity arises.

Consider now the second difference between pre-institution transformations and simulations. It is easy to see that translating models from $\mathcal{I}$ into $\mathcal{I}'$-model sets has the flavor of translating models from $\mathcal{I}'$ into $\mathcal{I}$ by a partial surjective function. Indeed the naturality of the model component of a pre-institution transformation guarantees that its counterimage has the partial-naturality property; a minor point is that to the counterimage be a function, the images of distinct models have to be disjoint; but until now no relevant counter-examples have been found where the images of distinct models are non-disjoint. So long simulations and transformations do not seem too far away, but the last point cannot be so easily disposed of; indeed the individual models in the image of a model M along a transformation are not required to satisfy the same sentences (under translation) as M, because the satisfaction invariant involves $\mathsf{Mo}_{\mathcal{T}}(M)$ as a whole. In fact there are transformations that cannot be expressed as simulations, but if in $\mathsf{Mo}_{\mathcal{T}}(M)$ a "canonical" representative can be found, satisfying the same sentences as M, then a simulation can be defined rephrasing the transformation.

Prop. 3.3.8 Let $\mathcal{I} = (\mathbf{Sign}, Sen, Mod, \models)$ and $\mathcal{I}' = (\mathbf{Sign}', Sen', Mod', \models')$ be *ps*-pre-institutions and $\mathcal{T}\colon \mathcal{I} \to \mathcal{I}'$ be a preinstitution transformation $(\mathsf{Si}_{\mathcal{T}}, \mathsf{Pr}_{\mathcal{T}}, \mathsf{Mo}_{\mathcal{T}})$ s.t.

- for every $\Sigma \in |\mathbf{Sign}|$ if $\mathsf{Mo}_{\mathcal{T}\Sigma}(M) \cap \mathsf{Mo}_{\mathcal{T}\Sigma}(M')$ is non-empty then $M = M'$;
- for every $M \in |Mod(\Sigma)|$ an $M' \in \mathsf{Mo}_{\mathcal{T}\Sigma}(M)$ exists *fully representing* M, i.e. s.t. $M \models_\Sigma E$ iff $M' \models'_{\mathsf{Si}_\mathcal{T}(\Sigma)} \mathsf{Pr}_\mathcal{T}(E)$ for all $E \in \mathsf{Pre}(\Sigma)$

Then $\mu: \wp(\mathcal{I}) \to \wp(\mathcal{I}')$ is a simulation, where $\mu_{Sign} = \mathsf{Si}_\mathcal{T}$, $\mu_{Sen} = \mathsf{Pr}_\mathcal{T}$ and μ_{Mod} is defined by:
$dom(\mu)_\Sigma = \{M' \mid M' \in \mathsf{Mo}_{\mathcal{T}\Sigma}(M), M'$ is fully representing some $M \in Mod(\Sigma)\}$ and if $M' \in \mathsf{Mo}_{\mathcal{T}\Sigma}(M)$, then $\mu_{Mod_\Sigma}(M') = M$.

Proof. The only non-trivial point is to show that μ_{Mod} is partially-natural; consider a signature morphism $\sigma: \Sigma \to \Sigma'$ and let M' belong to $dom(\mu)_{\Sigma'}$, i.e. M' fully represents M for some $M \in Mod(\Sigma')$ and $\mu_{Mod_{\Sigma'}}(M') = M$. Then $Mod'(\mathsf{Si}_\mathcal{T}(\sigma))(M') \in \mathsf{Mo}_{\mathcal{T}\Sigma}(Mod(\sigma)(M))$, because $\mathsf{Mo}_\mathcal{T}$ is a natural transformation; thus in order to show that $\mu_{Mod_\Sigma}(Mod'(\mathsf{Si}_\mathcal{T}(\sigma))(M')) = Mod(\sigma)(\mu_{Mod_{\Sigma'}}(M'))$ it is sufficient to prove that $Mod'(\mathsf{Si}_\mathcal{T}(\sigma))(M')$ fully represent $Mod(\sigma)(M) = Mod(\sigma)(\mu_{Mod_{\Sigma'}}(M'))$. Let E belong to $\mathsf{Pre}(\Sigma)$, then, by satisfaction condition, $Mod(\sigma)(M) \models_\Sigma E$ iff $M \models_{\Sigma'} \mathsf{Pre}(\sigma)(E)$, i.e., as M' fully represents M, iff $M' \models'_{\mathsf{Si}_\mathcal{T}(\Sigma')} \mathsf{Pr}_\mathcal{T}(\mathsf{Pre}(\sigma)(E))$, i.e., by satisfaction condition, iff $Mod'(\mathsf{Si}_\mathcal{T}(\sigma))(M') \models'_{\mathsf{Si}_\mathcal{T}(\Sigma)} \mathsf{Pr}_\mathcal{T}(E)$. ☐

Therefore any pre-institution transformation satisfying these properties is a simulation too; however notion like *adequacy* or *finitarity* (see [81]) are more naturally expressed for pre-institution transformations and hence working on the classical logical side, investigating for example compactness properties, may result easier in the preinstitution framework than using simulations. Moreover the equivalence between transformations providing a fully representative for every model and simulations follows from the *ps*-property of institutions and does not work on pre-institutions missing such property.

Note that the partial-naturality property is too weak to guarantee that $\mathcal{T}: \mathcal{I} \to \mathcal{I}'$ is a preinstitution transformation, where $\mathsf{Si}_\mathcal{T} = \mu_{Sign}$, $\mathsf{Pr}_\mathcal{T} = \wp \circ \mu_{Sen}$ and $\mathsf{Mo}_{\mathcal{T}\Sigma}(M) = \{M' \mid \mu_{Mod_\Sigma}(M') = M\}$, because the naturality of $\mathsf{Mo}_\mathcal{T}$ may be missing. Consider, indeed, a signature morphism $\sigma: \Sigma \to \Sigma'$ and let M' belong to $Mod(\Sigma')$; then $Mod'(\mu_{Sign}(\sigma))\mathsf{Mo}_{\mathcal{T}\Sigma'}(M')$ is a subset of $\mathsf{Mo}_{\mathcal{T}\Sigma}(Mod(\sigma)(M'))$, because μ_{Mod} is partially-natural, but non every $N \in \mathsf{Mo}_{\mathcal{T}\Sigma}(Mod(\sigma)(M')) = \{N \mid \mu_{Mod_\Sigma}(N) = Mod(\sigma)(M')\}$ is required to be the image along $Mod'(\mu_{Sign}(\sigma))$ of some

$$N' \in \mathsf{Mo}_{\mathcal{T}\Sigma'}(M') = \{N' \mid \mu_{Mod_{\Sigma'}}(N') = M'\},$$

so that $Mod'(\mu_{Sign}(\sigma))\mathsf{Mo}_{\mathcal{T}\Sigma'}(M')$ may be a proper subset of $\mathsf{Mo}_{\mathcal{T}\Sigma}(Mod(\sigma)(M'))$.

3.3.4 Institution Coding and Representations

The *institution coding* were presented in a draft paper [93] to investigate on the expressive power of LF (Edinburgh Logical Framework) and on "putting together" representation of logics in the common frame of the LF-institution. To simplify the presentation in [93], the simplest version of the concept of institution is adopted, which does not incorporate the notion of model morphism, so that sets (classes) of models are used instead of categories. Using the language of pre-institutions this corresponds to work on *ps* pre-institutions, i.e. on pre-institutions that preserve satisfaction.

Def. 3.3.9 Let $\mathcal{I} = (\mathbf{Sign}, Sen, Mod, \models)$ and $\mathcal{I}' = (\mathbf{Sign}', Sen', Mod', \models')$ be *ps* pre-institutions; then an *institution coding* $\rho: \mathcal{I} \to \mathcal{I}'$ consists of:

- a functor $\rho_{Sign}: \mathbf{Sign} \to \mathbf{Sign}'$;
- a natural transformation $\rho_{Sen}: Sen \to Sen' \circ \rho_{Sign}$;
- a natural transformation $\rho_{Mod}: Mod' \circ \rho_{Sign} \to Mod$.

An *institution representation* $\rho: \mathcal{I} \to \mathcal{I}'$ is an institution coding $\rho = (\rho_{Sign}, \rho_{Sen}, \rho_{Mod})$ s.t. the following *representation condition* holds
for any $\Sigma \in |\mathbf{Sign}|$, any $\phi \in Sen(\Sigma)$ and any $M \in Mod(\Sigma)$

$$M \models_\Sigma \phi \iff (\rho_{Mod\Sigma})^{-1}(M) {\models'}_{\rho_{Sign}(\Sigma)} \rho_{Sen}(\phi)$$

Note that if M does not belong to the image of ρ_{Mod}, i.e. $(\rho_{Mod\Sigma})^{-1}(M) = \emptyset$, then the representation condition requires that M satisfies all sentences in $Sen(\Sigma)$; thus the "non-represented" models do not carry relevant information from a logical point of view.

It is important to note that the representation condition capture the intuition that a model M is not represented by *any* model M' whose image along ρ_{Mod} is M but logically corresponds to its counterimage as whole class. Indeed also in the particular case of institution coding ρ s.t. ρ_{Mod} is surjective the representation condition is, in general, weaker than the following *strong representation condition*, that more closely resemble the satisfaction condition, because the strong representation condition relates the validity of individual models:
for any $\Sigma \in |\mathbf{Sign}|$, any $\phi \in Sen(\Sigma)$ and any $M' \in Mod'(\rho_{Sign}(\Sigma))$

$$\rho_{Mod\Sigma}(M') \models_\Sigma \phi \iff M' {\models'}_{\rho_{Sign}(\Sigma)} \rho_{Sen}(\phi)$$

The strong representation condition is too restrictive to capture many intuitively acceptable representations of logical systems; but if it holds for a surjective institution representation ρ, then ρ is a (total) simulation, too.

Since the representation condition involves classes of $\mathcal{I}'$-models, it would seem reasonable trying to apply a technique similar to the one proposed to relate simulations and pre-institution transformations, building an institution $\mathcal{I}'^{\wp}$ whose models are classes of $\mathcal{I}'$-models and where a class satisfy a sentence ϕ iff all its elements do satisfy ϕ. At this point, to make any institution representation $\rho: \mathcal{I} \to \mathcal{I}'$ a simulation $\mu: \mathcal{I} \to \mathcal{I}'^{\wp}$, two possible choices of μ_{Mod} seem natural: the total one where $\mu_{Mod} = \wp \circ \rho_{Mod}$ and the ρ-closed where if $C = {\rho_{Mod}}^{-1}(M)$, then $\mu_{Mod}(C) = M$, else $\mu_{Mod}(C)$ is undefined. But for the first one the satisfaction condition may be false, because classes of models that contain just pieces of the counterimage of $\mathcal{I}$-models are translated, so that the representation condition does not apply to them, and the second one fails to be partially-natural, because in general $Mod'(\rho_{Sign}(\sigma))({\rho_{Mod}}^{-1}(M'))$ is strictly contained in ${\rho_{Mod}}^{-1}(Mod(\sigma)(M'))$. Thus although the two notion seems strictly related and share most motivating examples, they are not the same. However, as in the case of transformations, many representations, providing for each $\mathcal{I}$-model a fully representative, can be rephrased as simulations.

Chapter 4

Translating Tools

The use of logic in computer science is experiencing vigorous growth. Since the applications are many, there are increasingly stronger interactions between the two fields that are having a profound impact on both of them. New logics are frequently been proposed, and new variants or adaptations of existing logics for new purposes are widespread.

This proliferation of logics—although certainly a sign of vitality and intellectual creativity—brings with it important conceptual challenges. In a sense, each logic is a different language and, as in the case of natural languages, there is often a serious need to bridge the gap between different languages by means of appropriate translations, and the danger of serious confusion when translations are not correct. There is also a related need to understand the essential features shared by logics in general so that systematic methods can be developed to deal with these problems.

It can be greatly advantageous to reuse entire logics, or key components of such logics. The advantages may be not only conceptual, although of course this is important; due to the existence of software systems supporting mechanized reasoning in a given logic, it may be possible to reuse a system developed for one logic—for example, a theorem-prover—to obtain a new system for another.

Translations between logics by appropriate mappings—especially if they are *conservative* in the sense of [63]—may provide a first way of reusing tools of one logic in another, by translating the appropriate sentences or proofs and using the original tool on the translations. This idea is here generalized to the case where entire components—for example the proof theory—of one of the logics involved may be completely missing, so that the appropriate mapping could not even be defined. The idea then is to borrow the missing components (as well as their associated tools if they exist) from a logic that has them in order to create, *ex* *nihilo* as it were, the full-fledged logic and tools that we desire. The relevant structure is transported using maps that only involve a limited aspect of the two logics in question—for example their model theory.

The constructions accomplishing this kind of borrowing of logical structure are very general and simple. Indeed they only depend upon a few abstract properties that hold under very general conditions given a pair of categories linked by adjoint functors. Therefore, the constructions capitalize on the fact that, as was shown in [63], the different components of a logic: entailment relation, model theory, and proof theory, are in a very precise technical sense *modular*, namely in that they can be added or deleted by means of constructions that are adjoint functors.

Specifically, it is shown that, given two institutions $\mathcal{I}$ and $\mathcal{I}'$ and a map of institutions $\mathcal{I} \longrightarrow \mathcal{I}'$, the entailment relation or the proof theory (or both things) can be borrowed from $\mathcal{I}'$ if they exist to use them for $\mathcal{I}$, and that completeness, if it holds, is preserved by the borrowing. Similarly, given two entailment systems $\mathcal{E}$ and $\mathcal{E}'$—i.e., the provability relations of two logics—and a map of entailment systems $\mathcal{E} \longrightarrow \mathcal{E}'$, it is shown that the proof theory can be borrowed from $\mathcal{E}'$ if it exists to endow $\mathcal{E}$ with a proof theory.

4.1 Introductory Examples

In this section the basic results in [8] are presented in the original form, in order to give some intuition of the general categorical construction presented in the next section. Thus the notion of inference system, and accordingly of soundness and completeness, are given in a rougher form than the corresponding notions for the entailment systems as presented in the next sections. It seems however useful to keep the original concreteness on one hand to get the intuition behind the general categorical construction illustrated here and on the other hand to state a weaker result than the categorical one, but with a larger spectrum of applicability.

According to the intuition that a simulation codes a new institution in terms of an old one, inference systems are translated backward via simulation; so that, starting from an inference system for $\mathcal{I}'$ and using a simulation $\mu: \mathcal{I} \rightarrow \mathcal{I}'$, a new system for $\mathcal{I}$ is built, which consists of: the preprocessing μ_{Sen} of the sentences of $\mathcal{I}$, coding them as sentences of $\mathcal{I}'$, followed by the application of the given system for $\mathcal{I}'$, and possibly by the postprocessing ${\mu_{Sen}}^{-1}$ to decode the results.

Def. 4.1.1 Let $\mathcal{I} = (\mathbf{Sign}, Sen, Mod, \models)$ be an institution and $\vdash$ be an inference system for $Sen(\Sigma)$, i.e. any relation $\vdash \subseteq \wp(Sen(\Sigma)) \times Sen(\Sigma)$. Then $\vdash$ is *sound* for $C \subseteq |Mod(\Sigma)|$ iff for every $\phi \in Sen(\Sigma)$ and every $\Gamma \subseteq Sen(\Sigma)$, $\Gamma \vdash \phi$ implies that for all $A \in C$ if $A \models_\Sigma \gamma$ for all $\gamma \in \Gamma$, then $A \models_\Sigma \phi$. If C is $|Mod(\Sigma)|$, then $\vdash$ is shortly said sound.

For every $\Psi \subseteq Sen(\Sigma)$ and every $C \subseteq |Mod(\Sigma)|$, the system $\vdash$ is *complete* w.r.t. Ψ and C iff for every $\psi \in \Psi$ and every $\Gamma \subseteq Sen(\Sigma)$

$$A \models_\Sigma \gamma \text{ for all } \gamma \in \Gamma \text{ implies } A \models_\Sigma \psi \quad \text{for every } A \in C$$

implies $\Gamma \vdash \psi$. If C is $|Mod(\Sigma)|$, then $\vdash$ is shortly said complete w.r.t. Ψ.

For every simulation $\mu\colon \mathcal{I} \to \mathcal{I}'$ and every inference system $\vdash'$ for $Sen'(\mu_{Sign}(\Sigma))$, the inference system $\vdash^\mu$ for $Sen(\Sigma)$ is defined by: $\Gamma \vdash^\mu \phi$ iff $\mu_{Sen}(\Gamma) \vdash' \mu_{Sen}(\phi)$. □

The definition of completeness as it stands is a generalization of the notion of completeness in algebraic frames; indeed, for examples, in the frame of (both partial and total) conditional specifications the *equational completeness* of a system $\vdash$ means that if an equation $t = t'$ holds in the model class of a set of conditional axioms Γ, then $\Gamma \vdash t = t'$. Thus the premises Γ are any set of sentences, while the consequence has to be an equation, i.e. a sentence in the selected subclass.

Note that if reflexivity, monotonicity and transitivity are required by the definition of inference system, as for the *entailment systems* of [63], then simulations preserve these properties, so that the translation of an entailment system is an entailment system, too.

The properties of simulations guarantee that if a system $\vdash'$ in the old institution $\mathcal{I}'$ is sound and complete w.r.t. the domain of the simulation μ, then the obtained system $\vdash^\mu$ is sound and complete for $\mathcal{I}$, too, as the following theorem shows.

Theorem 4.1.2 Let $\mathcal{I}$ and $\mathcal{I}'$ be institutions, $\mu\colon \mathcal{I} \to \mathcal{I}'$ be a simulation and $\vdash'$ be an inference system for $Sen'(\mu_{Sign}(\Sigma))$;

1. if $\vdash'$ is sound for $|dom(\mu)_\Sigma|$, then $\vdash^\mu$ is sound, too;

2. if $\vdash'$ is complete for $C' \subseteq |dom(\mu)_\Sigma|$ and $\Psi' \subseteq Sen'(\mu_{Sign}(\Sigma))$, then $\vdash^\mu$ is complete for $\mu_{Mod\Sigma}(C')$ and ${\mu_{Sen}}^{-1}(\Psi')$.

Proof.

1. Assume that $\Gamma \vdash^\mu \phi$, i.e. that $\mu_{Sen}(\Gamma) \vdash' \mu_{Sen}(\phi)$, and that $A \models_\Sigma \gamma$ for all $\gamma \in \Gamma$ for some $A \in |Mod(\Sigma)|$ and show that $A \models_\Sigma \phi$. Since $\mu_{Mod\Sigma}$ is surjective on the objects, there exists $A' \in |dom(\mu)_\Sigma|$ s.t. $\mu_{Mod\Sigma}(A') = A$ and hence $A' \models'_{\mu_{Sign}(\Sigma)} \mu_{Sen}(\gamma)$ for all $\gamma \in \Gamma$, due to the satisfaction condition. Thus, since $\vdash$ is sound for $dom(\mu)_\Sigma$ and $\mu_{Sen}(\Gamma) \vdash' \mu_{Sen}(\phi)$, $A' \models'_{\mu_{Sign}(\Sigma)} \mu_{Sen}(\phi)$. Therefore, due to the satisfaction condition, $A \models_\Sigma \phi$.

2. Let $\Gamma \subseteq Sen(\Sigma)$ and $\phi \in {\mu_{Sen}}_{\Sigma}^{-1}(\Psi)$ be s.t. $A \models \phi$ for every $A \in {\mu_{Mod}}_{\Sigma}(C')$ s.t. $A \models \gamma$ for all $\gamma \in \Gamma$. Then $A' \models' {\mu_{Sen}}_{\Sigma}(\phi)$ for every $A' \in C'$ s.t. $A' \models' {\mu_{Sen}}_{\Sigma}(\gamma)$ for all $\gamma \in \Gamma$, due to the satisfaction condition, i.e. $A' \models' \gamma'$ for all $\gamma' \in {\mu_{Sen}}_{\Sigma}(\Gamma)$ and hence, since $\vdash'$ is complete for C' and Ψ', ${\mu_{Sen}}_{\Sigma}(\Gamma) \vdash' {\mu_{Sen}}_{\Sigma}(\phi)$, i.e. $\Gamma \vdash^{\mu} \phi$. □

Note that if a system $\vdash'$ is sound for $Mod'(\mu_{Sign}(\Sigma))$, then it is sound for any of its subcategories and hence any general system for $\mathcal{I}'$ is sufficient, if only soundness matters; but if completeness is considered too, then the system $\vdash'$ is required to be complete for the domain of the simulation, which is a subclass of the whole model class, and hence in general does not need be even sound for the whole model class. Thus in general $\vdash'$ is not a general system for the new institution, but it is tailored to the simulation, contrary to the intuition that simulations translate general results from one formalism to another. However, if the domain of μ coincides with the model class of some set th' of sentences, and hence in particular if μ is logical, then starting from any sound and complete inference system w.r.t. the whole class of models, the above theorem can be applied to the system $\vdash'_{th'}$, defined by $\Phi \vdash'_{th'} \phi$ iff $\Phi \cup th \vdash' \phi$, thus recovering the desired level of generality. It is worth noting that this last construction is an instance of the categorical result presented in the next section, while the system $\vdash^{\mu}$ cannot be obtained in such a canonical way for the general case.

Cor. 4.1.3 Let $\mathcal{I}$ and $\mathcal{I}'$ be institutions, $\mu: \mathcal{I} \to \mathcal{I}'$ be a simulation s.t. $dom(\mu)_{\Sigma} = \{A' \mid A' \models'_{\mu_{Sign}(\Sigma)} \alpha', \alpha' \in th'\}$ for some $th' \subseteq Sen'(\mu_{Sign}(\Sigma))$ and $\vdash'$ be an inference system for $Sen'(\mu_{Sign}(\Sigma))$, which is sound and complete w.r.t. $\Psi' \subseteq Sen'(\mu_{Sign}(\Sigma))$. Then $\vdash^{\mu}_{th'}$ is sound and complete w.r.t. ${\mu_{Sen}}^{-1}(\Psi')$, where $\vdash_{th'}$ denotes the system defined by $\Phi' \vdash_{th'} \phi'$ iff $\Phi' \cup th' \vdash \phi'$.
Proof. Trivial instantiation of the Theorem 4.1.2. □

So far the main interest was on the translation of inference systems from the old into the new frame; however, note that soundness and completeness are preserved in the opposite direction, too.

4.2 Transporting Structures Across Categories

The transportation of structure that is studied here will involve two categories of logical structures, **C** and **D**, and functors $U: \mathbf{D} \to \mathbf{C}$ and $R: \mathbf{C} \to \mathbf{D}$ with R

right adjoint to U. In the applications presented here U, in spite of being a left adjoint, will have the flavor of a forgetful functor[1].

For the example presented in the last section, roughly speaking, $\mathbf{C}$ was a category of institutions, $\mathbf{D}$ was a category of logics, i.e. institutions endowed with a compatible inference system, U was the functor forgetting the deductive part and R was the functor adding the validity as (complete) inference system.

The basic construction applies to arbitrary categories $\mathbf{C}$ and $\mathbf{D}$ with functors U and R as above, and consists in the process of transporting to an object $c \in |\mathbf{C}|$ the structure of an object $d \in |\mathbf{D}|$ via a map $f: c \to U(d)$. The transported structure is enjoyed by an object $\tilde{c} \in |\mathbf{D}|$ which roughly speaking corresponds to c plus the features of d translated by f; formally it is the pullback of the following diagram, where η_d is the unity of the adjunction for the object d, and hence is strongly dependent on f and d.

$$\begin{tikzcd} R(c) \arrow[r, "R(f)"] & R(U(d)) \\ \tilde{c} \arrow[u, "j_{f,c}"] \arrow[r, "\tilde{f}"'] & d \arrow[u, "\eta_d"'] \end{tikzcd}$$

Moreover the construction is functorial, in the sense that preserves the f-consistent translations of c and d. Indeed it is a functor between two comma categories.

Theorem 4.2.1 Let $\mathbf{C}$ and $\mathbf{D}$ be categories and let $R: \mathbf{C} \to \mathbf{D}$ be the right adjoint of $U: \mathbf{D} \to \mathbf{C}$, with $\eta: 1_{\mathbf{D}} \to R \circ U$ the unit of the adjunction.

Denote by $\mathbf{D}'$ the comma category $\mathbf{D} \downarrow 1_{\mathbf{D}}$ (see e.g. [54]) and by $\mathbf{C}'$ the comma category $\mathbf{C} \downarrow U$.

1. Let $U': \mathbf{D}' \to \mathbf{C}'$ be the functor defined as follows:
   - $U'(d: D \to D') = (U(d): U(D) \to U(D'), D')$ for every object $(d: D \to D') \in |\mathbf{D}'|$;
   - $U'(f, f') = (U(f), f')$ for every arrow $(f, f') \in \mathbf{D}'((d_1: D_1 \to D'_1), (d_2: D_2 \to D'_2))$.

[1]This is of course some what counterintuitive, but it is partly explained by the fact that in most examples $U \circ R$ is the identity on $\mathbf{C}$ with the co-unity $\epsilon_C = Id_C$ for every object $C \in \mathbf{C}$.

2. If for every object $C \in |\mathbf{C}|$, every object $D' \in |\mathbf{D}|$ and every arrow $c \in \mathbf{C}(C, U(D'))$ the pullback of $R(c)$ and $\eta_{D'}$ exists, then there is a functor $R': \mathbf{C}' \to \mathbf{D}'$, defined as follows:

   - for each object $(c: C \to U(D'), D') \in |\mathbf{C}'|$ the object $R'(c, D') = (\tilde{c}: \tilde{C} \to D')$ is defined by the following pullback:

$$\begin{tikzcd} R(C) \arrow[r, "R(c)"] & R \circ U(D') \\ \tilde{C} \arrow[u, "j_{c,D'}"] \arrow[r, "\tilde{c}"'] & D' \arrow[u, "\eta_{D'}"'] \end{tikzcd}$$

   - for each arrow (g, g') in $\mathbf{C}'$ from $(c_1: C_1 \to U(D'_1), D'_1)$ into $(c_2: C_2 \to U(D'_2), D'_2)$ the arrow $R'(g, g') = (g^\sharp, g')$, where $g^\sharp$ is defined as the unique arrow in $\mathbf{D}'$ into the pullback $\tilde{c}_2$ s.t. both

     (a) $j_{c_2,D'_2} \circ g^\sharp = R(g) \circ j_{c_1,D'_1}$;

     (b) $\tilde{c}_2 \circ g^\sharp = g' \circ \tilde{c}_1$.

   If such a functor R' exists, then $\mathbf{C}$ is said to *admit generalization under R and U*.

3. Under the hypothesis of 2, R' is the right adjoint of U'.

Proof.

1. It is immediate to see that U' is a functor, because it is basically the componentwise application of U.

2. For each $(g, g') \in \mathbf{C}'((c_1: C_1 \to U(D'_1), D'_1), (c_2: C_2 \to U(D'_2), D'_2))$ the existence of such a $g^\sharp$ is shown first. For this purpose, given that $(\tilde{c}_2, j_{c_2,D'_2})$, is the pullback of $R(c_2)$ and $\eta_{D'_2}$, it is sufficient to show that $R(c_2) \circ R(g) \circ j_{c_1,D'_1} = \eta_{D'_2} \circ g' \circ \tilde{c}_1$.

   Consider the following diagram:

$$\begin{tikzcd}
R(C_1) \arrow[rr, "R(c_1)"] \arrow[dr, "R(g)"] & & R \circ U(D'_1) \arrow[dr, "R \circ U(g')"] & \\
& R(C_2) \arrow[rr, "R(c_2)"] & & R \circ U(D'_2) \\
\widetilde{C_1} \arrow[uu, "j_{c_1,D'_1}"] \arrow[rr, "\tilde{c_1}"] & & D'_1 \arrow[uu, "\eta_{D'_1}"] \arrow[dr, "g'"] & \\
& \widetilde{C_2} \arrow[uu, "j_{c_2,D'_2}"] \arrow[rr, "\tilde{c_2}"] & & D'_2 \arrow[uu, "\eta_{D'_2}"]
\end{tikzcd}$$

The front and the back faces of this cube are commutative, by definition of $\widetilde{C_2}$ and $\widetilde{C_1}$ as pullbacks; the upper side is commutative, too, since (g, g') is an arrow from c_1 into c_2 and finally the right side is commutative, because η is a natural transformation from the identity functor into $R \circ U$. Therefore, all paths on the cube from $\widetilde{C_1}$ into $R \circ U(D'_2)$ are equal and hence, in particular $R(c_2) \circ R(g) \circ j_{c_1,D'_1} = \eta_{D'_2} \circ g' \circ \tilde{c_1}$.

Therefore, since $\tilde{c_2}$ is the pullback of $R(c_2)$ and $\eta_{D'_2}$, there exists a unique $g^\sharp$ s.t. both

(a) $j_{c_2,D'_2} \circ g^\sharp = R(g) \circ j_{c_1,D'_1}$;

(b) $\tilde{c_2} \circ g^\sharp = g' \circ \tilde{c_1}$.

The last property guarantees that $(g^\sharp, g')$ is an arrow from $\tilde{c_1}$ into $\tilde{c_2}$. Moreover, since obviously $(Id_{\widetilde{C}}, Id_{D'})$ and $(\bar{g}^\sharp \circ g^\sharp, \bar{g}' \circ g')$ satisfy the above conditions respectively for the identity and the composition, the uniqueness guarantees that $Id^\sharp_{(c:C \to U(D'),D')} = Id_{\tilde{c}:\widetilde{C} \to D'}$ and $(\bar{g} \circ g)^\sharp = \bar{g}^\sharp \circ g^\sharp$, so that R' preserves identities and compositions and hence it is a functor.

3. To show that U' is the left adjoint of R', an arrow $\alpha_{\mathsf{X}}: \mathsf{X} \to R' \circ U'(\mathsf{X})$ for each object $\mathsf{X} = (x: X \to X')$ in $\mathbf{D'}$ is first defined and then $(R' \circ U'(\mathsf{X}), \alpha_{\mathsf{X}})$ is shown to be a universal object from X to R'.

Consider any object $\mathsf{X} = (x : X \to X')$ in $\mathbf{D}'$, then $R' \circ U'(\mathsf{X}) = (\widetilde{U(x)} : \widetilde{U(X)} \to X')$ is defined by the following pullback diagram.

$$
\begin{tikzcd}
R \circ U(X) \arrow[r, "R \circ U(x)"] & R \circ U(X') \\
\widetilde{U(X)} \arrow[u, "j_{U(x),X'}"] \arrow[r, "\widetilde{U(x)}"'] & X' \arrow[u, "\eta_{X'}"']
\end{tikzcd}
$$

Since η is a natural transformation from the identity functor into $R \circ U$, $R \circ U(x) \circ \eta_X = \eta_{X'} \circ x$ and hence, by $\widetilde{U(x)}$ being a pullback, there exists a unique $\alpha_{\mathsf{X}} : X \to \widetilde{U(X)}$ commuting the following diagram

$$
\begin{tikzcd}
 & & X \arrow[r, "x"] \arrow[d, dashed, "\alpha_{\mathsf{X}}"'] \arrow[dr, "x"] \arrow[ddll, "\eta_X"'] & X' \arrow[d, "Id_{X'}"] \\
 & & \widetilde{U(X)} \arrow[r, "\widetilde{U(x)}"] \arrow[dll, "j_{U(x),X'}"] & X' \arrow[dll, "\eta_{X'}"] \\
R \circ U(X) \arrow[r, "R \circ U(x)"'] & R \circ U(X') & &
\end{tikzcd}
$$

i.e., such that

(a) $\eta_X = j_{U(x),X'} \circ \alpha_{\mathsf{X}}$;

(b) $x = \widetilde{U(x)} \circ \alpha_{\mathsf{X}}$.

The second condition guarantees that the pair $(\alpha_{\mathsf{X}}, Id_{X'})$, from now on abbreviated by α_{X}, is an arrow from X to $R' \circ U'(\mathsf{X})$ in $\mathbf{D}'$.

To show that $(R' \circ U'(\mathsf{X}), \alpha_{\mathsf{X}})$ is universal, it is sufficient to prove that for every object $\mathsf{Y} = (y : C \to U(D'), D')$ in $\mathbf{C}'$ and every arrow $\mathsf{f} = (f, f') : \mathsf{X} \to R'(\mathsf{Y})$ there exists a unique $\bar{k} : U'(\mathsf{X}) \to \mathsf{Y}$ s.t. $R'(\bar{k}) \circ \alpha_{\mathsf{X}} = \mathsf{f}$.

$$\begin{tikzcd}
\mathsf{X} \arrow[r, "\alpha_{\mathsf{X}}"] \arrow[dr, "\mathsf{f}"'] & R' \circ U'(\mathsf{X}) \arrow[d, dotted, "R'(\bar{k})"] & U'(\mathsf{X}) \arrow[d, "\bar{k}"] \\
& R'(\mathsf{Y}) & \mathsf{Y}
\end{tikzcd}$$

Consider the following diagram, that is commutative, because it is obtained by pasting together the two commutative diagrams given by the pullback definition of $\widetilde{C}$ and by f being an arrow from X to $R'(\mathsf{Y})$ in $\mathbf{D}'$.

$$\begin{tikzcd}
X \arrow[r, "f"] \arrow[d, "x"'] & \widetilde{C} \arrow[r, "j_{y,D'}"] \arrow[d, "\widetilde{y}"] & R(C) \arrow[d, "R(y)"] \\
X' \arrow[r, "f'"'] & D' \arrow[r, "\eta_{D'}"'] & R \circ U(D')
\end{tikzcd}$$

Since U is the left adjoint of R, there exists a unique $k_f \colon U(X) \to C$ s.t. $j_{y,D'} \circ f = R(k_f) \circ \eta_X$, and there exists a unique $g_f \colon U(X) \to U(D')$ s.t. $R(y) \circ j_{y,D'} \circ f = R(g_f) \circ \eta_X$.

In order to show that $\bar{k} = (k_f, f')$ is an arrow from $U'(\mathsf{X})$ into Y in $\mathbf{C}'$, i.e. that $U(f') \circ U(x) = y \circ k_f$, it is sufficient to show that both sides satisfy the equation required for g_f.
Since (f, f') is an arrow in $\mathbf{D}'$, $R(U(f') \circ U(x)) \circ \eta_X = R(U(\widetilde{y} \circ f)) \circ \eta_X$, and, since η is a natural transformation, $R(U(\widetilde{y} \circ f)) \circ \eta_X = \eta_{D'} \circ \widetilde{y} \circ f$. Moreover, by definition of $\widetilde{y}$, $\eta_{D'} \circ \widetilde{y} \circ f = R(y) \circ j_{y,D'} \circ f$ and hence $g_f = U(f') \circ U(x)$. Moreover $R(y \circ k_f) \circ \eta_X = R(y) \circ j_{y,D'} \circ f$ by definition of k_f and hence $g_f = y \circ k_f$, too, so that $U(f') \circ U(x) = y \circ k_f$.

It is shown that $\mathsf{f} = R'(\bar{k}) \circ \alpha_{\mathsf{X}}$, i.e. that

- $f = k_f^\sharp \circ \alpha_{\mathsf{X}}$;
- $f' = f' \circ Id_{X'}$

The second property is trivial; to show the first, because of the uniqueness of factorization through the pullback $\widetilde{C}$, it is sufficient to show the two equalities

- $j_{y,D'} \circ f = j_{y,D'} \circ (k_f^\sharp \circ \alpha_{\mathsf{X}})$;

- $\widetilde{y} \circ f = \widetilde{y} \circ (k_f^\sharp \circ \alpha_{\mathsf{X}})$

Because of condition 2a, $j_{y,D'} \circ k_f^\sharp \circ \alpha_{\mathsf{X}} = R(k_f) \circ j_{U(x),X'} \circ \alpha_{\mathsf{X}}$ and, because of condition 3a, $R(k_f) \circ j_{U(x),X'} \circ \alpha_{\mathsf{X}} = R(k_f) \circ \eta_X$; finally, by definition of k_f, $R(k_f) \circ \eta_X = j_{y,D'} \circ f$.

Because of condition 2b, $\widetilde{y} \circ k_f^\sharp \circ \alpha_{\mathsf{X}} = f' \circ \widetilde{U(x)} \circ \alpha_{\mathsf{X}}$ and, because of condition 3b, $f' \circ \widetilde{U(x)} \circ \alpha_{\mathsf{X}} = f' \circ x$; finally, since (f, f') is an arrow in $\mathbf{D}'$, $f' \circ x = \widetilde{y} \circ f$.

It is shown that $\bar{k}$ is the unique arrow that makes the diagram commute. Assume that $\mathsf{f} = R'(\bar{h}) \circ \alpha_{\mathsf{X}}$ for some $\bar{h} = (h, h')$ and prove that $\bar{h} = \bar{k}$. It is immediate to see that $h' \circ Id_{X'} = f' = k' \circ Id_{X'}$, so that $h' = k' = f'$.

Since $f = h^\sharp \circ \alpha_{\mathsf{X}}$, $j_{y,D'} \circ f = j_{y,D'} \circ h^\sharp \circ \alpha_{\mathsf{X}}$ and, because of 2a, $j_{y,D'} \circ h^\sharp \circ \alpha_{\mathsf{X}} = R(h) \circ j_{x,X'} \circ \alpha_{\mathsf{X}}$.
Moreover, because of 3a, $R(h) \circ j_{x,X'} \circ \alpha_{\mathsf{X}} = R(h) \circ \eta_X$, so that, by definition of k_f, $h = k_f$. □

It is also worth noting that the naturality of η is sufficient for the first two statements and that the adjoint situation between U and R is needed just to get that U' and R' are adjoint, too. In the next sections this general construction will be applied to the particularly interesting cases of building logics and logical systems starting from institution maps and of building proof calculi starting from maps of entailment systems.

4.3 General Logics

Since the significant examples of application of the general categorical construction are all largely based on the concepts of general logic (see [63]), this section is devoted to recall the basic definitions and results of this theory that were not presented in Sects. 3.3 and 1.2; for a detailed discussion of these concepts see [63].

Putting together an institution and a compatible entailment system, i.e. an entailment system that is sound w.r.t. the institution, a *logic* is obtained, where tools to deal with inference and model theoretic aspects are both at hand. But, since entailment systems disregard the computational aspects, the concept of (structured) proofs is not formalized; thus *proof calculi* are introduced to cover also this feature.

Def. 4.3.1 A *logic* (see [63], def. 6) is a 5-tuple $\mathcal{L} = (\mathbf{Sign}, Sen, Mod, \models, \vdash)$ such that:

1. $(\mathbf{Sign}, Sen, \vdash)$ is an entailment system;

2. $(\mathbf{Sign}, Sen, Mod, \models)$ is an institution, and

3. the following *soundness* condition is satisfied: for every $\Sigma \in |\mathbf{Sign}|$, $\Gamma \subseteq Sen(\Sigma)$ and $\phi \in Sen(\Sigma)$, $\Gamma \vdash_\Sigma \phi \Rightarrow \Gamma \models_\Sigma \phi$.

A *proof calculus* (see [63], def. 12) is a 6-tuple

$$\mathcal{P} = (\mathbf{Sign}, Sen, \vdash, P, \mathrm{Pr}, \pi)$$

with:

1. $(\mathbf{Sign}, Sen, \vdash)$ is an entailment system;

2. $P\colon \mathbf{Th}_0 \to \mathbf{Struct}_\mathcal{P}$ a functor; for each theory T, the object $P(T) \in \mathbf{Struct}_\mathcal{P}$ is called its *proof-theoretic structure*;

3. $\mathrm{Pr}\colon \mathbf{Struct}_\mathcal{P} \to \mathbf{Set}$ a functor; for each theory T, the set $\mathrm{Pr}(P(T))$ is called its *set of proofs*. Then *proofs* will denote the composite functor $\mathrm{Pr} \circ P\colon \mathbf{Th}_0 \to \mathbf{Set}$;

4. $\pi\colon proofs \Rightarrow Sen$ a natural transformation, such that for each theory $T = (\Sigma, \Gamma)$, the image of $\pi_T\colon proofs(T) \to Sen(T)$ is the set $\Gamma^\bullet$ of all sentences ϕ s.t. $\Gamma \vdash \phi$.

A *logical system* (see [63], def. 12) is a 8-tuple

$$\mathcal{S} = (\mathbf{Sign}, Sen, Mod, \vdash, \models, P, \mathrm{Pr}, \pi)$$

with:

1. $(\mathbf{Sign}, Sen, Mod, \models, \vdash)$ is a logic;

2. $(\mathbf{Sign}, Sen, \vdash, P, \mathrm{Pr}, \pi)$ is a proof calculus. □

The crucial point of any categorical approach is the choice of arrows between objects more than the objects themselves. In [63] the maps of entailment systems, institutions, logics, proof calculi and logical systems are discussed in detail and their definitions are carefully motivated; here just the definitions are presented.

Def. 4.3.2 Given entailment systems $\mathcal{E} = (\mathbf{Sign}, Sen, \vdash)$, $\mathcal{E}' = (\mathbf{Sign}', Sen', \vdash')$, a *map of entailment systems* $(\Phi, \alpha)\colon \mathcal{E} \to \mathcal{E}'$ consists of a natural transformation

$\alpha\colon Sen \Rightarrow Sen' \circ \Phi$ and α-sensible functor $\Phi\colon \mathbf{Th}_0 \to \mathbf{Th}'_0$ satisfying the following property:

$$\Gamma \vdash_\Sigma \phi \text{ implies } \alpha_\Sigma(\Gamma) \vdash'_{\Phi(\Sigma)} \alpha_\Sigma(\phi),$$

where for every theory $th' = (\Sigma', \Delta')$ in $\mathbf{Th}'_0$, every $\Delta' \subseteq Sen'(\Sigma')$ and every $\phi' \in Sen'(\Sigma')$ the notation $\Gamma' \vdash'_{th'} \phi'$ stands for $\Gamma' \cup \Delta' \vdash'_{\Sigma'} \phi'$.

Given logics $\mathcal{L} = (\mathbf{Sign}, Sen, Mod, \models, \vdash)$, $\mathcal{L}' = (\mathbf{Sign}', Sen', Mod', \models', \vdash')$, a *map of logics* $(\Phi, \alpha, \beta)\colon \mathcal{L} \to \mathcal{L}'$ is a map of the underling institution s.t. $(\Phi, \alpha)\colon ent(\mathcal{L}) \to ent(\mathcal{L}')$ is a map of the underlying entailment systems, too.

Given proof calculi $\mathcal{P} = (\mathbf{Sign}, Sen, \vdash, P, \mathrm{Pr}, \pi)$ and $\mathcal{P}' = (\mathbf{Sign}', Sen', \vdash', P', Pr', \pi')$, a *map of proof calculi* $(\Phi, \alpha, \gamma)\colon \mathcal{P} \to \mathcal{P}'$ consists of a map $(\Phi, \alpha)\colon ent(\mathcal{P}) \to ent(\mathcal{P}')$ of the underlying entailment systems together with a natural transformation $\gamma\colon proofs \Rightarrow proofs' \circ \Phi$ such that $\pi'_\Phi \circ \gamma = \alpha \circ \pi$.

Given logical systems $\mathcal{S} = (\mathbf{Sign}, Sen, Mod, \vdash, \models, P, \mathrm{Pr}, \pi)$ and $\mathcal{S}' = (\mathbf{Sign}', Sen', Mod', \vdash', \models', P', Pr', \pi')$, a *map of logical systems*

$$(\Phi, \alpha, \beta, \gamma)\colon \mathcal{S} \to \mathcal{S}'$$

consists of a map of the underling logics $(\Phi, \alpha, \beta)\colon log(\mathcal{S}) \to log(\mathcal{S}')$ and a map of the underlying proof calculi $(\Phi, \alpha, \gamma)\colon pcalc(\mathcal{S}) \to pcalc(\mathcal{S}')$. □

The relationships among the categories of entailment systems, institutions, logics, proof calculi, and logical systems are clearly illustrated by the following diagram, where the arrows are forgetful functors[2]

$$\begin{tikzcd}
 & & \underline{LogSys} \arrow[dl, "log"'] \arrow[dr, "pcalc"] & \\
 & \underline{Log} \arrow[dl, "inst"'] \arrow[dr, "ent"] & & \underline{PCalc} \arrow[dl, "ent_{PC}"] \\
\underline{Inst} & & \underline{Ent} &
\end{tikzcd}$$

It is also interesting to note that both *log* and ent_{PC} have a right adjoint, corresponding to regard theorems as proofs of themselves; moreover *inst* has both a left and a right adjoint, obtained by adding respectively the subset and the validity relations as entailment, and finally *ent* has a right adjoint, too, whose construction uses comma categories to build models and was sketched in Sect. 1.2.5.

[2]In [63] a more complex diagram is presented, where also *subcalculi* are considered in order to be able to capture effectiveness and efficiency considerations

4.4 Building Logics

Since there are different ways of translating institutions (and hence logics), there are also several possibilities of building a logic on top of an institution $\mathcal{I}$ using an already known logic $\mathcal{L}'$ and a translation of $\mathcal{I}$ into the institution underlying $\mathcal{L}'$, by applying the construction introduced in the above section. The most promising case is using *maps of institutions* (see e.g. [63]), because completeness is preserved and the entailment system $\vdash$ defined for $\mathcal{I}$ by the pullback can be easily described as the coding of the theories via a map and the application of the entailment system $\vdash'$ of $\mathcal{L}'$, so that any (finitary) description of $\vdash'$ is also a description of $\vdash$. Indeed the entailment system $\vdash$ coincides with the system $\vdash^{\mu}_{th'}$ introduced in the first section.

Prop. 4.4.1 Denote by $\underline{Log}$ the category of logics with maps of logics as arrows and by $\underline{Inst}$ the category of institutions with maps of institutions as arrows. Then the functor $(\_)^+ : \underline{Inst} \to \underline{Log}$, defined by

- $(\mathcal{I})^+ = (\mathbf{Sign}, Sen, Mod, \models, \vdash_{\models})$, where $\Gamma \vdash_{\models} \phi$ iff $A \models \phi$ for every model A s.t. $A \models \gamma$ for all $\gamma \in \Gamma$, for every institution $\mathcal{I} = (\mathbf{Sign}, Sen, Mod, \models)$ and

- $(\rho)^+ = \rho$ for every map ρ,

is the right adjoint of the forgetful functor $inst : \underline{Log} \to \underline{Inst}$, defined by $inst(\mathcal{L}) = (\mathbf{Sign}, Sen, Mod, \models)$ for every logic $\mathcal{L} = (\mathbf{Sign}, Sen, Mod, \models, \vdash)$ and $inst(\rho) = \rho$ for every map ρ.

Moreover the unity of the adjunction is the (family of the) embedding of $(\mathbf{Sign}, Sen, Mod, \models, \vdash)$ into $(\mathbf{Sign}, Sen, Mod, \models, \vdash_{\models})$.

Proof. Proposition 31 of [63]. $\square$

Although in general $\underline{Log}$ does not have pullbacks, they do exist in the particular case of pulling back the unit of the adjunction along a generic map.

Lemma 4.4.2 Let $\mathcal{I} = (\mathbf{Sign}, Sen, Mod, \models)$ be an institution, $\mathcal{L}'$ a logic and $\rho : \mathcal{I} \to inst(\mathcal{L}')$ a map of institutions. Denoting by $\eta_{\mathcal{L}'}$ the embedding of $\mathcal{L}'$ into $(inst(\mathcal{L}'))^+$, the following diagram is a pullback, where $\mathcal{L} = (\mathbf{Sign}, Sen, Mod, \models, \vdash)$

and $\vdash$ is defined by: $\Gamma \vdash_\Sigma \phi$ iff both $\alpha_\Sigma(\Gamma) \vdash'_{\Phi(\Sigma,\emptyset)} \alpha_\Sigma(\phi)$ and $\Gamma \models_\Sigma \phi$.

$$\begin{tikzcd}
(\mathcal{I})^+ \arrow[r, "(\rho)^+"] & (inst(\mathcal{L}'))^+ \\
\mathcal{L} \arrow[u, "(Id_{\mathbf{Sign}}, Id_{Sen}, Id_{Mod})"] \arrow[r, "(\rho)^+"'] & \mathcal{L}' \arrow[u, "\eta_{\mathcal{L}'}"']
\end{tikzcd}$$

Proof. Routine checks. $\square$

Because of the above Prop. 4.4.1 and Lemma 4.4.2, the Theorem 4.2.1 applies and for every map ρ between institutions builds the entailment system that in [8] was denoted by $\vdash^\rho_{th}$ for $th = ax(\Phi(\Sigma))$.

Prop. 4.4.3 The category $\underline{Inst}$ admits generalization under $(\_)^+$ and the forgetful functor *inst*.

Proof. By Theorem 4.2.1, that applies because of Prop. 4.4.1 and Lemma 4.4.2. $\square$

For every map of institution from $\mathcal{I}$ into $U(\mathcal{L}')$, the entailment system built for $\mathcal{I}$ can be informally described by:

- coding sentences by means of the map;
- using the entailment system of $\mathcal{L}'$;
- checking the soundness of the deduction.

Moreover, if the component of the map dealing with the models is surjective, the last step is not needed. Indeed for every model $A \in |Mod(\Sigma, \Gamma)|$ there exists (at least) a model $A' \in |Mod'(\Phi(\Sigma, \Gamma))|$ s.t. $\beta_\Sigma(A') = A$ so that $A \models_\Sigma \phi$ iff $A' \models'_{sign(\Phi(\Sigma))} \alpha_\Sigma(\phi)$. Thus $\Gamma \models_\Sigma \phi$ iff $\Phi(\Sigma, \Gamma) \models'_{sign(\Phi(\Sigma))} \alpha_\Sigma(\phi)$ and hence, because of the soundness of $\vdash'$, if $\alpha_\Sigma(\Gamma) \vdash'_{\Phi(\Sigma,\emptyset)} \alpha_\Sigma(\phi)$, then $\Phi(\Sigma, \Gamma) \models'_{sign(\Phi(\Sigma))} \alpha_\Sigma(\phi)$ and hence $\Gamma \models_\Sigma \phi$. Therefore, an effective way of generating the entailment system of $\mathcal{L}'$, if any, can also be used to check whether $\Gamma \vdash \phi$.

It is also worth pointing out that completeness is preserved by this construction, as the following proposition shows.

Prop. 4.4.4 Using the notation of Lemma 4.4.2, if $\mathcal{L}'$ is complete, then $\mathcal{L}$ is complete, too.

Proof. Indeed assume to have for some $\Gamma \subseteq Sen(\Sigma)$ and $\psi \in Sen(\Sigma)$ that $A \models \psi$ for all $A \in |Mod(\Sigma)|$ s.t. $A \models \gamma$ for all $\gamma \in \Gamma$. Then, validity being both preserved and reflected by maps, $A' \models' \alpha(\psi)$ for all $A' \in |Mod'(\Phi(\Sigma))|$ s.t. $A' \models' \alpha(\gamma)$ for all $\gamma \in \Gamma$ and hence, $\vdash'$ being complete, $\alpha(\Gamma) \vdash'_{\Phi(\Sigma)} \alpha(\psi)$. Thus, by definition, $\Gamma \vdash_\Sigma \psi$. □

Example 4.4.5 Consider again the reduction of many-sorted equational Horn-clause logic to one-sorted Horn-clause logic by the logical simulation (and hence surjective map of logic) μ^M.

As entailment system of the logic $\mathcal{L}_\mathcal{L}$ having the one-sorted institution as underline institution, the entailment generated by (a version[3] of) the classical Birkhoff's deductive system has been chosen.

For every one-sorted signature $\Sigma = (Op, P)$ and every set Γ of conditional sentences on Σ, $\Gamma \vdash_{B\Sigma} \phi$ iff ϕ is in the inductive closure of Γ and the following axioms qualifying the equality, where t, t', t'', t_i, t'_i are terms on Op, $p \in P_n$ and $f \in Op_n$:

$$\begin{array}{l}
t = t \\
t = t' \supset t' = t \\
t = t' \wedge t' = t'' \supset t = t'' \\
t_1 = t'_1 \wedge \ldots \wedge t_n = t'_n \wedge p(t_1, \ldots, t_n) \supset p(t'_1, \ldots, t'_n) \\
t_1 = t'_1 \wedge \ldots \wedge t_n = t'_n \supset f(t_1, \ldots, t_n) = f(t'_1, \ldots, t'_n)
\end{array}$$

w.r.t. the following inference rules of weakening, instantiation and modus ponens.

$$\textit{weakening} \qquad \frac{\epsilon_1 \wedge \ldots \wedge \epsilon_n \supset \epsilon}{\epsilon_1 \wedge \ldots \wedge \epsilon_n \wedge \eta_1 \wedge \ldots \wedge \eta_k \supset \epsilon}$$

$$\textit{instantiation} \qquad \frac{\phi}{\phi[t_1/x_1 \ldots t_n/x_n]} \quad t_i \text{ terms of the same sort as } x_i$$

$$\textit{modus ponens} \qquad \frac{\epsilon_1 \wedge \ldots \wedge \epsilon_n \supset \epsilon, \eta_1 \wedge \ldots \wedge \eta_k \supset \epsilon_i}{\epsilon_1 \wedge \ldots \wedge \epsilon_{i-1} \wedge \eta_1 \wedge \ldots \wedge \eta_k \wedge \epsilon_{i+1} \wedge \ldots \wedge \epsilon_n \supset \epsilon}$$

Since $\vdash_B$ is defined as an inductive closure, obviously it is monotonic, transitive and reflexive; moreover it is easy to check from the definition that it also satisfies the $\vdash$-translation condition and that it is sound, so that $\mathcal{L}_\mathcal{L} = (\mathbf{Sign}_\mathcal{L}, Sen_\mathcal{L}, Mod_\mathcal{L}, \models_\mathcal{L}, \vdash_B)$ is a logic.

Then, by applying the above theorem, an entailment system $\vdash$ for $\mathcal{MS}$ is defined by:

[3]the weakening rule is usually not included in the definition of the classical Birkhoff system, but it is needed to achieve a system complete w.r.t. *conditional* sentences. Moreover this system works on conditional formulas built on both equalities and predicates.

for every many-sorted signature $\Sigma = (S, F)$ and every set Γ of conditional sentences on Σ,

$$\Gamma \vdash_\Sigma V.t_1 = t'_1 \wedge \ldots \wedge t_n = t'_n \supset t = t'$$

(for $V(x_i) = s_i$ for $i = 1 \ldots k$ and $V(x)$ undefined otherwise) iff

$$x_1 : s_1 \wedge \ldots \wedge x_k : s_k \wedge t_1 = t'_1 \wedge \ldots \wedge t_n = t'_n \supset t = t'$$

is in the inductive closure of the following axioms, where t, t', t'', t_i, t'_i are terms on Op_n for $Op_n = \cup_{s_1,\ldots,s_n,s \in S} F_{s_1 \ldots s_n,s}$, $f \in Op_n$ and $op \in F_{s_1 \ldots s_k,s}$:

$$\begin{array}{l}
x_1 : s_1 \wedge \ldots \wedge x_k : s_k \supset op(x_1, \ldots, x_k) : s \\
x_1 : s_1 \wedge \ldots \wedge x_k : s_k \wedge t_1 = t'_1 \wedge \ldots \wedge t_n = t'_n \supset t = t' \\
t = t \\
t = t' \supset t' = t \\
t = t' \wedge t' = t'' \supset t = t'' \\
t = t' \wedge t : s \supset t' : s \\
t_1 = t'_1 \wedge \ldots \wedge t_n = t'_n \supset f(t_1, \ldots, t_n) = f(t'_1, \ldots, t'_n)
\end{array}$$

w.r.t. the inference rules of weakening, instantiation and modus ponens.

Since $\vdash_B$ is complete w.r.t. conditional sentences, as it is provable by standard techniques, this completeness is inherited by $\vdash$, because of Prop. 4.4.4; thus the logic $\mathcal{L}_{\mathcal{MS}} = (\mathbf{Sign}_{\mathcal{MS}}, Sen_{\mathcal{MS}}, Mod_{\mathcal{MS}}, \models_{\mathcal{MS}}, \vdash)$ is complete for the many-sorted institution. $\Box$

Some more applications may be found in [60], where translations of partial, of Horn-clauses and of order-sorted logics in terms of equational type logic are presented in order to use the ET-inference system and the connected rewrite tools; although these translations are not formalized as logical simulations, they can be so and the results obtained by their applications are an instance of the ones presented here.

Consider now simulations as arrows between institutions and entailment-preserving simulations as arrows between logics. In this case, since the translation of models is partial, the condition $\Gamma \models \phi$ does not imply that $\mu_{Sen}(\Gamma) \models' \mu_{Sen}(\phi)$ for some simulation $\mu: \mathcal{I} \to \mathcal{I}'$, because an $\mathcal{I}'$-model may exist satisfying $\mu_{Sen}(\Gamma)$ but for which $\mu_{Sen}(\phi)$ does not hold, provided that such a model does not belong to the simulation domain. Thus adding to any institution the validity relation as entailment system does not give a functor. Since this is the only reasonable way to define a right adjoint of the forgetful functor from logics into institutions (that is also a left inverse, at the least), the general construction does not apply.

It is also worth pointing out that considering morphisms of institutions (see e.g. [44]) as arrows, requiring that entailment is preserved in the case of logics, the forgetful functor does have a right adjoint (adding the minimal entailment relation), so that also in this case the institutions category admits generalization. The entailment built by this construction can be informally described by $\Gamma \vdash \phi$ iff $\Gamma \models \phi$ and there are Γ' and ϕ' s.t. $\alpha(\Gamma') = \Gamma$, $\alpha(\phi') = \phi$ and $\Gamma' \vdash' \phi'$. Thus $\vdash$ is in general of no use, because it requires to "invent" the Γ' and ϕ' or to check a usually infinite set of possibility.

4.4.1 Applications

Recently higher-order specifications have become a standard tool in algebraic specifications, with a particular interest in the specification of partial higher-order functions. Higher-order functional spaces can be handled using the usual first-order algebraic specifications (see e.g. [66]), by restricting the signatures (S, F) to the ones where S is a subset of a set of functional sorts s.t. for every $(s_1 \times \ldots \times s_n \to s_{n+1}) \in S$ an explicit application operator belongs to the signature; moreover the models are required to be extensional, i.e. two elements of a functional sort yielding the same result on every input have to be equal. From a logical point of view, in the total case (see [62, 77]) an equationally complete system for the higher-order models may be obtained by enriching any (first-order) equationally complete system by the rule

$*$
$$\frac{f(x_1, \ldots, x_n) = g(x_1, \ldots, x_n)}{f = g}$$

where f, g are terms of sort $(s_1 \times \ldots \times s_n \to s_{n+1})$ and each x_i is a variable of sort s_i not appearing in f and g.

Instead in the partial case the above rule $*$ is insufficient to achieve a complete system as the following example shows[4].

Fact 4.4.6 Let *FSp* be a conditional higher-order specification. Then the system, from now on denoted by $US(FSp)^\star$, consisting of all the axiom schemas and

[4] It is interesting noting that $US(PSp)$ is complete w.r.t. any positive conditional specification PSp (and in general $US(sp)$ is not w.r.t. non-positive sp), but the same counter-example applies to the system $CL_v(sp)$ introduced in the sequel, that is seq-complete w.r.t. conditional specifications. The point is that, due to the logical equivalence between strong equalities and disjunctions (of a particular form), it is not necessary to deduce the equality of the application of two functions to general terms representing a generic input in order to get their identity.

inference rules of $US(FSp)$ (see Sects. 2.4, 2.5) and of the following additional inference rule:

$\star$
$$\frac{f(x_1, ..., x_n) = g(x_1, ..., x_n)}{f = g}$$

where $f, g \in T_\Sigma(X - \{x_1, \dots, x_n\})_{|s_1 \times \dots \times s_n \to s}$, $x_i \in X_{s_i}$
is not complete for EMod(FSp) and the empty family of variables.
Proof. Consider the following specification

spec FSp_4 =
  sorts $s, (s \to s)$
  constants
    f, g of type $(s \to s)$
    e of type s
  axioms
    α_1 $D(f) \wedge f{=}g \supset D(e)$
    α_2 $D(f(x)) \supset D(e)$
    α_3 $D(g(x)) \supset D(e)$
    α_4 $D(g)$
    α_5 $D(f)$

Then e is defined in each model A of FSp_4; indeed either there exists an element a s.t. $f^A(a)$ or $g^A(a)$ is defined, and in this case, because of α_2 and α_3, also e^A is defined, or both f^A and g^A are defined, because of α_4 and α_5, and their result over any possible assignment is undefined so that, because of the extensionality, $f^A = g^A$ and hence $D(e)$ follows from α_1. But it easy to check that $US(FSp)^\star \not\vdash D(e)$. $\square$

However there is a logical simulation, i.e. a surjective map of institution, based on a skolemization procedure of higher-order by strongly conditional partial algebras, so that Props. 4.4.3 and 4.4.4 apply and hence an equationally complete system for the higher-order models may be simulated by any equationally complete system for strongly conditional partial models. The intuition of this construction is that for each couple f, g of distinct functional elements a *witness* of their difference, i.e. an input (tuple) a s.t. $f(a) \neq g(a)$, exists; thus it is sufficient to introduce function symbols to denote the witnesses.

Def. 4.4.7 Denoting by $\mathcal{PAR}_{Fin}$ the subinstitution of $\mathcal{PAR}$ whose sentences are finitary, let $\mu^E\colon \mathcal{PHO} \to \mathcal{PAR}$ ($\mu^E\colon \mathcal{FPHO} \to \mathcal{PAR}_{Fin}$) be the simulation consisting of:

- ${\mu^E}_{Sign}$: $\mathbf{Sign}_{\mathcal{PHO}} \to \mathbf{Sign}_{\mathcal{PAR}}$ is defined by ${\mu^E}_{Sign}(S,F) = (S', F')$, where $S = S'$ and

$$F' = F \cup_{s=(s_1 \times \ldots \times s_n \to s_{n+1}) \in S} \{x_{s,i}: s \times s \to s_i \mid i = 1, \ldots, n\}$$

and ${\mu^E}_{Sign}(\sigma, \phi) = (\sigma, \phi')$, where $\phi'(f) = \phi(f)$ for all $f \in F$ and $\phi(x_{s,i}) = x_{\sigma(s),i}$.

- ${\mu^E}_{Sen}$: $Sen_{\mathcal{PHO}} \to Sen_{\mathcal{PAR}} \circ {\mu^E}_{Sign}$ is the natural transformation whose components are embedding.

- ${\mu^E}_{Mod}$: $Mod_{\mathcal{PAR}} \circ {\mu^E}_{Sign} \to Mod_{\mathcal{PHO}}$ is defined by

  - $dom(\mu^E)_\Sigma$ is the full subcategory of $Mod_{\mathcal{PAR}}({\mu^E}_{Sign}(\sigma, \phi))$ whose objects are the partial algebras A which satisfy the set $th(\mu^E)$ of axioms

  $$f(x_{s,1}(f,g), \ldots, x_{s,n}(f,g)) = g(x_{s,1}(f,g), \ldots, x_{s,n}(f,g)) \supset f = g$$

  for all $(s_1 \times \ldots \times s_n \to s_{n+1}) \in S$

  - Let $\iota: \Sigma \to {\mu^E}_{Sign}(\Sigma)$ be the signature embedding; then ${\mu^E}_{Mod}(A') = Mod_{\mathcal{PHO}}(\iota)(A')$ and ${\mu^E}_{Mod}(h') = Mod_{\mathcal{PHO}}(\iota)(h')$. In the sequel $A'_{|\Sigma}$ denotes $Mod_{\mathcal{PHO}}(\iota)(A')$ and $h'_{|\Sigma}$ denotes $Mod_{\mathcal{PHO}}(\iota)(h')$. □

Since ${\mu^E}_{Sen}$ is the identity and ${\mu^E}_{Mod}$ is a family of forgetful functors, it is quite easy to check that μ^E is a simulation; the only non-trivial step is to check that ${\mu^E}_{Mod}$ is surjective on the objects, i.e. that for every extensional algebra A an expansion A' of A exists (i.e. an algebra A' s.t. $A'_{|\Sigma} = \mathsf{A}$) s.t. $A' \in |dom(\mu^E)_\Sigma|$. Hence it is sufficient to define $x^{A'}_{s,i}$ on A in such a way that A' satisfies the axioms

$$f(x_{s,1}(f,g), \ldots, x_{s,n}(f,g)) = g(x_{s,1}(f,g), \ldots, x_{s,n}(f,g)) \supset f = g$$

Since A is extensional, for all $s = s_1 \times \ldots \times s_n \to s_{n+1}$ and all $\phi, \psi \in s^{\mathsf{A}}$ either $\phi = \psi$ or there exist $a_i \in s_i^{\mathsf{A}}$ for $i = 1 \ldots n$ s.t. $\phi(a_1, \ldots, a_n) \neq \psi(a_1, \ldots, a_n)$; in the first case let $x^{A'}_{s,i}(\phi, \psi)$ be undefined for $i = 1 \ldots n$, in the second one let $x^{A'}_{s,i}(\phi, \psi)$ be such an a_i for $i = 1 \ldots n$. Then it is easy to check that $A' \in dom(\mu^E)_\Sigma$.

In order to get an equationally complete inference system for the partial higher-order case by the application of the general pullback construction for the logical simulation μ^E, an equationally complete system for the partial strongly conditional case has to be exhibit first.

Since the intuition behind the system and the completeness proof are non-trivial, the next section is devoted to the introduction of the conditional system $CL_v(sp)$, that is obtained by adding just one new rule to the system $US(sp)$ of the Chapter 2, Sect. 2.4.2.

The $CL_v(sp)$ system

In order to get some intuition about the needed generalization of the system $US(sp)$, consider the following specification sp having free models for all families of variables but s.t. $\mathsf{Fr}(US(sp), \emptyset)$ is not a model of sp.

spec sp_9 =
  sorts s_1, s_2
  opns
    $a, b \colon \to s_1$
    $e \colon \to s_2$
  axioms
    α_1 $D(a) \supset D(e)$
    α_2 $a{=}b \supset D(e)$
    α_3 $D(b) \supset D(e)$

In the models of sp_9, $D(a)$ or $D(b)$ or $a{=}b$ holds, by definition of strong equality; so that because of α_1, α_2 and α_3 also $D(e)$ has to hold in the models of sp_9, while $US(sp_9) \not\vdash D(e)$. Moreover for all families X of variables the free object $(Fr(X), m)$ for X in $PMod(sp_9)$ is defined by:

Algebra $Fr(X)$ =
  ${s_1}^{Fr(X)} = X_{s_1}$
  ${s_2}^{Fr(X)} = X_{s_2} \cup \{\bullet\}$
  $a^{Fr(X)}, b^{Fr(X)}$ are undefined
  $e^{Fr(X)} = \bullet$
  $m(x) = x$

The specification sp_9 suggests that, to make $US(sp)$ complete, a rule $\star$ is needed, having basically the form

$$\star \qquad \frac{(\Delta_1 \cup \{D(t)\}) \supset \epsilon, (\Delta_2 \cup \{D(t')\}) \supset \epsilon, (\Delta_3 \cup \{t{=}t'\}) \supset \epsilon}{(\Delta_1 \cup \Delta_2 \cup \Delta_3) \supset \epsilon}$$

where t and t' are ground terms. If t and t' are not ground, the above rule has obviously to be generalized by keeping track of the variables in t and t' in the way introduced in Sect. 2.4.2.

However, since the logic considered here is infinitary, the above rule $\star$ has to be generalized also to eliminate an infinite number of premises in one step.

Just in order to capture some intuition about the required generalization, first consider a finitary case where there are more than one strong equalities to eliminate (even though every finitary case can be handled by a finite number of applications of $\star$; see Sect. 4.4.1 below).

spec sp_{10} =
  sorts s_1, s_2
  opns
    $a, b, c, d \colon \to s_1$
    $e \colon \to s_2$
  axioms
    $\alpha_1 \quad D(a) \wedge D(c) \supset D(e)$
    $\alpha_2 \quad a{=}b \wedge D(c) \supset D(e)$
    $\alpha_3 \quad D(b) \wedge D(c) \supset D(e)$
    $\alpha_4 \quad D(a) \wedge D(d) \supset D(e)$
    $\alpha_5 \quad a{=}b \wedge D(d) \supset D(e)$
    $\alpha_6 \quad D(b) \wedge D(d) \supset D(e)$
    $\alpha_7 \quad D(a) \wedge c{=}d \supset D(e)$
    $\alpha_8 \quad a{=}b \wedge c{=}d \supset D(e)$
    $\alpha_9 \quad D(b) \wedge c{=}d \supset D(e)$

In all models of sp_{10} at least one among $D(a)$ or $D(b)$ or $a{=}b$ holds, by definition of strong equality, and analogously at least one among $D(c)$ or $D(d)$ or $c{=}d$ holds. Therefore in all models of sp_{10} the premises of at least one among $\alpha_1 \ldots \alpha_9$ hold and hence $D(e)$ holds in the models of sp_{10}. □

Note that in all models of sp_{10} the premises of at least one axiom hold since $\{prem(\alpha_i) \mid i = 1 \ldots 9\}$ is the set $\{D(a), D(b), a{=}b\} \times \{D(c), D(d), c{=}d\}$ and one among $\{D(a), D(b), a{=}b\}$ and one among $\{D(c), D(d), c{=}d\}$ has to hold. Then for a generic finitary case from a family $\{\phi_j \mid j = 1 \ldots n\}$ of conditional formulas an elementary formula ϵ is deduced iff:

- $cons(\phi_j) = \epsilon$ for all $j = 1 \ldots n$;
- $\{prem(\phi_j) \mid j = 1 \ldots n\}$ is the set

  $$\{D(t_1), D(t'_1), t_1{=}t'_1\} \times \ldots \times \{D(t_m), D(t'_m), t_m{=}t'_m\}$$

  for some t_i, t'_i and $i = 1 \ldots m$.

Indeed in all models A one among $D(t_i), D(t'_i), t_i{=}t'_i$ holds for all $i = 1 \ldots m$ and hence there exists $j \in \{1 \ldots n\}$ s.t. $A \models \delta$ for all $\delta \in prem(\phi_j)$.

The point is that there is a set $\{\phi_j \mid j = 1 \ldots n\}$ of conditional formulas with the same consequence and s.t. the premises of the ϕ_j cannot be simultaneously falsified, because any choice of one element in every $prem(\phi_j)$ is always true, containing the set $\{D(t), D(t'), t{=}t'\}$.

The above discussion is summarized in a formal statement.

Def. 4.4.8 Let $\Gamma = \{\Gamma_j \mid j \in J\}$ be a possibly non-countable set of sub-sets of $\text{EForm}(\Sigma, Var)$.

- $\mathsf{Sec}(\Gamma)$ denotes the set of all *sections* of Γ, where a section of Γ is a subset of $\text{EForm}(\Sigma, Var)$ of the form $\{\gamma_j \mid j \in J\}$ for some $\gamma_j \in \Gamma_j$ for all $j \in J$;
- Γ is *ininfluent* iff for all $\Psi \in \mathsf{Sec}(\Gamma)$ there exist $t, t' \in T_\Sigma(Var)$ s.t. $D(t), D(t'), t = t' \in \Psi$.

For $sp = (\Sigma, Ax)$, the inference system $CL_v(sp)$ (where the index v stands for "variables") consists of the axioms and inference rules of $US(sp)$ and of the following inference rule, where Θ_j, Γ_j are arbitrary countable subsets of $\text{EForm}(\Sigma, Var)$, $\epsilon \in \text{EForm}(\Sigma, Var)$:

Elimination $$\frac{\{\Theta_j \cup \Gamma_j \supset \epsilon \mid j \in J\}}{D(\cup_{j\in J} Var(\Gamma_j)) \cup (\cup_{j\in J}\Theta_j) \supset \epsilon} \qquad \{\Gamma_j \mid j \in J\} \text{ is ininfluent.}$$ □

The intuition that $\Gamma = \{\Gamma_j \mid j \in J\}$ is ininfluent iff Γ is, in some sense, $\{D(t_1), D(t'_1), t_1{=}t'_1\} \times \ldots \times \{D(t_m), D(t'_m), t_m{=}t'_m\} \times \ldots$ is formalized by the following proposition, that states that Γ is ininfluent iff its sections are subsets of the sections of a family of sets of the form $\{D(t), D(t'), t{=}t'\}$.

Prop. 4.4.9 Let $\Gamma = \{\Gamma_j \mid j \in J\}$ be any family of sets of elementary formulas. Then Γ is ininfluent iff there exist a set I, either $\{1 \ldots k\}$ or $\mathbb{N}$, and terms t_i, t'_i for $i \in I$ s.t. for every section σ of $\mathcal{R} = \{R(t_i, t'_i) \mid i \in I\}$, where $R(t, t') = \{D(t), D(t'), t = t'\}$, there exists $j \in J$ s.t. $\Gamma_j \subseteq \sigma$.

Moreover if J and Γ_j for all $j \in J$ are finite, then I is finite too.

Proof.

$\Rightarrow$ Consider the set C of all possible couples of terms appearing in Γ; then C is denumerable at the most; fix an enumeration for C, so that $C = \{(t_i, t'_i) \mid i \in I\}$, where I is $\{1 \ldots k\}$ if C is finite (in particular if J and Γ_j for all $j \in J$ are finite), otherwise $I = \mathbb{N}$.

Assume by contradiction that a section σ of $\mathcal{R} = \{R(t_i, t'_i) \mid i \in I\}$ exists s.t. $\Gamma_j \subseteq \sigma$ does not hold for every $j \in J$, i.e. there exists $\gamma_j \in \Gamma_j$ s.t. $\gamma_j \notin \sigma$ for every $j \in J$. Thus, by definition, $\Psi = \{\gamma_j \mid j \in J\}$ is a section for Γ and $\Psi \cap \sigma = \emptyset$. Because of the ininfluence of Γ, there exists $n \in I$ s.t. $R(t_n, t'_n) \subseteq \Psi$, so that $\Psi \cap \sigma \neq \emptyset$, as σ is a section of $\mathcal{R}$, contrary to the construction of Ψ.

$\Leftarrow$ If $\Gamma_j = \emptyset$ for some $j \in J$, then Γ is obviously ininfluent; otherwise let Ψ be a section of Γ and assume by contradiction that there do not exist terms t, t' s.t. $R(t,t') \subseteq \Psi$. So in particular for every $i \in I$ there exists $\sigma_i \in R(t_i, t'_i)$ s.t. $\sigma_i \notin \Psi$ and hence $\Psi \cap \sigma = \emptyset$, where σ is the section $\{\sigma_i \mid i \in I\}$ of $\mathcal{R}$. By hypothesis there exists $j \in J$ s.t. $\emptyset \neq \Gamma_j \subseteq \sigma$ and hence, Ψ being a section of Γ, $\Psi \cap \sigma \neq \emptyset$, contrary to the construction of σ. □

Prop. 4.4.10 The inference system $CL_v(sp)$ is a system for sp.

Proof. Since Prop. 2.4.10 proves that $US(sp)$ is a system for sp, it is sufficient to show that the elimination rule is sound.

Let A belong to $PMod(sp)$ and $A \models_V \theta$ for some valuation V for

$$Var(D(Var(\cup_{j\in J}\Gamma_j)) \cup (\cup_{j\in J}\Theta_j) \supset \epsilon)$$

and all $\theta \in (D(Var(\cup_{j\in J}\Gamma_j)) \cup (\cup_{j\in J}\Theta_j))$, and show that $A \models_V \epsilon$. To this end it is sufficient to prove that there exists $j \in J$ s.t. $A \models_V \theta$ for all $\theta \in \Theta_j \cup \Gamma_j$, because $A \models (\Theta_j \cup \Gamma_j) \supset \epsilon$ for all $j \in J$ by inductive hypothesis, and V is also a valuation for $Var(\Theta_j \cup \Gamma_j \supset \epsilon)$ in A.

Since $A \models_V \theta$ for all $\theta \in \cup_{j\in J}\Theta_j$, it is sufficient to show that there exists $j \in J$ s.t. $A \models_V \gamma$ for all $\gamma \in \Gamma_j$.

By contradiction assume that for every $j \in J$, there exists $\gamma_j \in \Gamma_j$ s.t. $A \not\models_{\mathcal{PAR}_V} \gamma_j$. Let Ψ be the section $\{\gamma_j \mid j \in J\}$; then, by definition of Ψ, $A \not\models_{\mathcal{PAR}_V} \psi$ for all $\psi \in \Psi$, which contradicts the assumption that for all $\Psi \in \mathsf{Sec}(\{\Gamma_j \mid j \in J\})$ there exist $t, t' \in T_\Sigma(Var)_{|s}$ s.t. $D(t), D(t'), t{=}t' \in \Psi$, because if $A \not\models_{\mathcal{PAR}_V} D(t)$ and $A \not\models_{\mathcal{PAR}_V} D(t')$, then $A \models_V t{=}t'$. □

Remark. Note that, because of rules 2b and 5 of $CL_v(sp)$, for all terms t, t', all sets Δ of elementary formulas and all elementary formulas ϵ

$$CL_v(sp) \vdash \{t{=}t'\} \cup \Delta \supset \epsilon \iff CL_v(sp) \vdash \{t'{=}t\} \cup \Delta \supset \epsilon$$

and analogously

$$CL_v(sp) \vdash \Delta \supset t{=}t' \iff CL_v(sp) \vdash \Delta \supset t'{=}t$$

i.e. $t{=}t'$ and $t'{=}t$ are interchangeable and hence in the sequel "$\ldots t{=}t' \ldots$" is regarded as an abbreviation for " …both $t{=}t'$ and $t'{=}t\ldots$" , or for " …$t{=}t'$ or $t'{=}t\ldots$" and of similar phrases. □

Note that the elimination rule is clearly a generalization of rule $\star$ given above. Now an example of use of elimination rule in an infinitary case is shown.

spec sp_9 =
  sorts s_1, s_2, s_3
  opns
    $a_1: \to s_1$
    $a_2: \to s_2$
    $a_3: \to s_3$
    $f_1, g_1: s_1 \to s_1$
    $f_2, g_2: s_2 \to s_2$
  axioms
    $\alpha_1 \quad \{f_1{}^n(a_1) = g_1{}^n(a_1) \mid n \in \mathbb{N}\} \supset D(a_3)$
    $\alpha_2 \quad \{f_2{}^m(a_2) = g_2{}^m(a_2) \mid m \in \mathbb{N}\} \supset D(a_3)$
    $\alpha_3 \quad D_{s_1}(x_1) \wedge D_{s_2}(x_2) \supset D(a_3)$

From α_3, by the instantiation rule, for all $i, j \in \mathbb{N}$ the following sentences follow:

$$\theta^1{}_{i,j} = (D(f_1{}^i(a_1)) \wedge D(f_2{}^j(a_2)) \supset D(a_3))$$

$$\theta^2{}_{i,j} = (D(f_1{}^i(a_1)) \wedge D(g_2{}^j(a_2)) \supset D(a_3))$$

$$\theta^3{}_{i,j} = (D(g_1{}^i(a_1)) \wedge D(f_2{}^j(a_2)) \supset D(a_3))$$

$$\theta^4{}_{i,j} = (D(g_1{}^i(a_1)) \wedge D(g_2{}^j(a_2)) \supset D(a_3))$$

$\Gamma = \{prem(\alpha_1)\} \cup \{prem(\alpha_2)\} \cup \{prem(\theta^k{}_{i,j}) \mid k = 1 \ldots 4; i, j \in \mathbb{N}\}$ is ininfluent and hence from α_1, α_2 and $\{prem(\theta^k{}_{i,j}) \mid k = 1 \ldots 4, i, j \in \mathbb{N}\}$, by the elimination rule, $D(a_3)$ is deduced. Indeed let Ψ be a section of Γ; by definition, in Ψ there are an element of $prem(\alpha_1)$, say $f_1{}^m(a_1) = g_1{}^m(a_1)$, an element of $prem(\alpha_2)$, say $f_2{}^n(a_2) = g_2{}^n(a_2)$, and an element of $prem(\theta^k{}_{n,m})$ for every $k = 1 \ldots 4$ and hence $\{D(f_1{}^m(a_1)), D(g_1{}^m(a_1))\} \subseteq \Psi$ or $\{D(f_2{}^n(a_2)), D(g_2{}^n(a_2))\} \subseteq \Psi$, by definition of $prem(\theta^k{}_{n,m})$. Therefore $\{D(f_1{}^m(a_1)), D(g_1{}^m(a_1)), f_1{}^m(a1) = g_1{}^m(a_1)\} \subseteq \Psi$ or $\{D(f_2{}^n(a_2)), D(g_2{}^n(a_2)), f_2{}^n(a_2) = g_2{}^n(a_2)\} \subseteq \Psi$. $\square$

The $CL_g(sp)$ System

In order to show the seq-completeness of $CL_v(sp)$, the simpler deduction system $CL_g(sp)$ is introduced first, where the elimination rule has been restricted to work only on ground formulas, and it is shown that the seq-completeness of $CL_g(sp)$ for sp and the empty family of variables implies the seq-completeness of $CL_v(sp)$ for sp and every family of variables; to do this in particular in a preliminar lemma deduction in $CL_v(sp)$ is proved to reduce to ground deduction in $CL_g(sp)$. Then $CL_g(sp)$ is proved to be seq-complete for sp and the empty family of variables and hence $CL_v(sp)$ is seq-complete.

Def. 4.4.11 Let sp be (Σ, Ax); the inference system $CL_g(sp)$ (where the index g stands for "ground") consists of the axioms and inference rules of $US(sp)$ and of the elimination rule restricted to work on sentences without variables, i.e. of the following inference rule, where Θ_j, Γ_j are arbitrary countable subsets of $\text{EForm}(\Sigma, \emptyset)$, $\epsilon \in \text{EForm}(\Sigma, \emptyset)$:

$$\textit{Ground Elimination} \qquad \frac{\{\Theta_j \cup \Gamma_j \supset \epsilon \mid j \in J\}}{(\cup_{j \in J}\Theta_j) \supset \epsilon} \qquad \{\Gamma_j \mid j \in J\} \text{ is ininfluent}$$

The seq-completeness of $CL_g(sp)$ w.r.t. the empty family of variables and sp will be called *gseq-completeness*, and analogously eeq-completeness of $CL_g(sp)$ w.r.t. the empty family of variables and sp will be called *geeq-completeness.* □

For the proof of both the gseq-completeness theorem and some of the intermediate results it is crucial to show that $CL_g(sp)$ satisfies the deduction theorem, which is well known and fundamental in classical logic, w.r.t. elementary ground formulas, i.e. for all sets of elementary ground formulas Γ

$$CL_g(\Sigma, Ax \cup \Gamma) \vdash \Delta \supset \epsilon \iff CL_g(sp) \vdash \Gamma' \cup \Delta \supset \epsilon \text{ for some } \Gamma' \subseteq \Gamma$$

Prop. 4.4.12 [*Deduction Theorem*] Let sp be the conditional specification (Σ, Ax) and Γ be a set of elementary ground formulas over Σ.

Then $CL_g(\Sigma, Ax \cup \Gamma) \vdash \Delta \supset \epsilon$ iff $CL_g(sp) \vdash \Gamma' \cup \Delta \supset \epsilon$, for some $\Gamma' \subseteq \Gamma$.

Proof. Let sp' be the conditional specification $(\Sigma, Ax \cup \Gamma)$.

$\Leftarrow$ Assume that $CL_g(sp) \vdash \Gamma' \cup \Delta \supset \epsilon$. Then, by definition of sp and sp', $CL_g(sp') \vdash \Gamma' \cup \Delta \supset \epsilon$, too. Moreover $CL_g(sp') \vdash \gamma$ for all $\gamma \in \Gamma'$, since $\Gamma' \subseteq \Gamma$, and hence $CL_g(sp') \vdash \Delta \supset \epsilon$ follows too, by Modus Ponens rule, since Γ' is a set of ground formulas.

$\Rightarrow$ The proof is by induction over the definition of $CL_g(sp')$.

Proper axioms

- If $(\Delta \supset \epsilon) \in Ax$, then $CL_g(sp) \vdash \Delta \supset \epsilon$ and hence the thesis follows for $\Gamma' = \emptyset$;
- If $(\Delta \supset \epsilon) \in \Gamma$, then $\Delta = \emptyset$; now it is shown that $CL_g(sp) \vdash \epsilon \supset \epsilon$ for every ground elementary formula ϵ, and hence the thesis follows for $\Gamma' = \{\epsilon\}$.
  - * Let ϵ be the formula $D(t)$, with $t \in T_{\Sigma|s}$. Then $CL_g(sp) \vdash D(t) \wedge t{=}t \supset D(t)$ by rule 4 and $CL_g(sp) \vdash t{=}t$ by rule 2a; so from rule 5 $CL_g(sp) \vdash D(t) \supset D(t)$ follows too.

* Let ϵ be the formula $t{=}t'$, with $t, t' \in T_{\Sigma|s}$. Then $CL_g(sp) \vdash t{=}t' \supset t'{=}t$ and $CL_g(sp) \vdash t'{=}t \supset t{=}t'$ by rule 2b ; so, by rule 5, $CL_g(sp) \vdash t{=}t' \supset t{=}t'$ follows too.

Axioms 2…4 obvious, for $\Gamma' = \emptyset$.

Modus Ponens Assume that $CL_g(sp') \vdash \Delta \cup \Xi \supset \epsilon$ and $CL_g(sp') \vdash \Delta_\xi \supset \xi$ for all $\xi \in \Xi$ so that $CL_g(sp')$ deduces

$$D(Var(\Xi) - Var(\Delta \cup \cup_{\xi\in\Xi}\Delta_\xi \supset \epsilon)) \cup \Delta \cup \cup_{\xi\in\Xi}\Delta_\xi \supset \epsilon.$$

By inductive hypothesis, $CL_g(sp) \vdash \Delta \cup \Xi \cup \Gamma'' \supset \epsilon$ and $CL_g(sp) \vdash \Delta_\xi \cup \Gamma'_\xi \supset \xi$ for all $\xi \in \Xi$ and some Γ'', $\Gamma'_\xi \subseteq \Gamma$. Thus because of rule 5

$$CL_g(sp) \vdash D(Y) \cup \Delta \cup \Gamma'' \cup (\cup_{\xi\in\Xi}\Delta_\xi) \cup (\cup_{\xi\in\Xi}\Gamma'_\xi) \supset \epsilon$$

where $Y = Var(\Xi) - Var(\Delta \cup \Gamma'' \cup (\cup_{\xi\in\Xi}\Delta_\xi) \cup (\cup_{\xi\in\Xi}\Gamma'_\xi) \supset \epsilon)$ i.e. $Y = Var(\Xi) - Var(\Delta \cup (\cup_{\xi\in\Xi}\Delta_\xi) \supset \epsilon)$, because both Γ'' and Γ'_ξ are sets of ground formulas,; thus the thesis follows for $\Gamma' = \Gamma'' \cup \cup_{\xi\in\Xi}\Gamma'_\xi$.

Instantiation/Abstraction Assume that $CL_g(sp') \vdash \Delta \supset \epsilon$ and denote by γ^* the formula $\gamma[t_x/x \mid x \in X_s, s \in S]$; so that

$$CL_g(sp') \vdash \{D(t_x) | x \in X_s, s \in S\} \cup \{\delta^* | \delta \in \Delta\} \supset \epsilon^*.$$

By inductive hypothesis $CL_g(sp) \vdash \Gamma'' \cup \Delta \supset \epsilon$, with $\Gamma'' \subseteq \Gamma$. Thus, by rule 6,

$$CL_g(sp) \vdash \{D(t_x) \mid x \in X_s, s \in S\} \cup \{\delta^* \mid \delta \in \Delta\} \cup \{\gamma^* \mid \gamma \in \Gamma''\} \supset \epsilon^*$$

i.e. , since every $\gamma \in \Gamma''$ is a ground formula,

$$CL_g(sp) \vdash \{D(t_x) \mid x \in X_s, s \in S\} \cup \Gamma'' \cup \{\delta^* \mid \delta \in \Delta\} \supset \epsilon^*$$

and hence the thesis follows for $\Gamma' = \Gamma''$.

Elimination Assume that (i) $CL_g(sp') \vdash \Delta_j \cup \Gamma_j \supset \epsilon$ for all $j \in J$ and that (ii) there exist $t, t' \in T_\Sigma$ s.t. $D(t), D(t')$ and $t{=}t'$ belong to Ψ for all $\Psi \in \mathsf{Sec}(\{\Gamma_j \mid j \in J\})$, so that $CL_g(sp) \vdash (\cup_{j\in J}\Delta_j) \supset \epsilon$. Because of (i) and of the inductive hypothesis

$$CL_g(sp) \vdash \Gamma'_j \cup \Gamma_j \cup \Delta_j \supset \epsilon,$$

with $\Gamma'_j \subseteq \Gamma$ for all $j \in J$. Thus, by (ii) and elimination rule, $CL_g(sp) \vdash (\cup_{j\in J}\Gamma'_j) \cup (\cup_{j\in J}\Delta_j) \supset \epsilon$, i.e. the thesis follows for $\Gamma' = \cup_{j\in J}\Gamma'_j$. □

Remark. Note that it is immaterial which variables are used in deduction both by $CL_g(sp)$ and $CL_v(sp)$, because the α-rule of the λ-calculus can be derived by rules 5, 6 and 1. Therefore in the following the variables used during the proof of any theorem by $CL_g(sp)$ are assumed w.l.o.g. not to be in X. $\square$

Lemma 4.4.13 Let $\Sigma = (S, F)$ be a signature, $sp = (\Sigma, Ax)$ be a conditional specification and X be an S-sorted family of variables.

Let sp_X denote the conditional specification (Σ_X, Ax_X), where $\Sigma_X = (S, F \cup \{op_x \colon \to s \mid x \in X_s\}_{s \in S})$ and $Ax_X = Ax \cup \{D(op_x) \mid x \in X\}$, and ϕ^* denote $\phi[op_x/x \mid x \in X]$ for all conditional formulas ϕ.

For all conditional formulas $\Delta \supset \epsilon$, $CL_g(sp_X) \vdash \Delta^* \supset \epsilon^*$ implies $CL_v(sp) \vdash D(X) \cup \Delta \supset \epsilon$.

Proof. Since rule 6 allows to arbitrary increase the definedness assertions in the premises of any deduced formula, it is sufficient to prove that $CL_g(sp_X) \vdash \Delta^* \supset \epsilon^*$ implies $CL_v(sp) \vdash D(X') \cup \Delta \supset \epsilon$ for some $X' \subseteq X$. The proof is done by induction on the definition of $CL_g(sp_X)$.

Proper axioms

- If $(\Delta \supset \epsilon) \in Ax$, then $CL_v(sp) \vdash \Delta \supset \epsilon$, by definition.
- Otherwise $(\Delta \supset \epsilon) = D(op_x)$ and the thesis follows by rule 1.

Axioms 2...4 obvious.

Modus Ponens Assume that $CL_g(sp_X) \vdash \Theta^* \cup \Gamma^* \supset \epsilon^*$ and $CL_g(sp_X) \vdash {\Theta_\gamma}^* \supset \gamma^*$ for all $\gamma \in \Gamma$; then

$$CL_g(sp_X) \vdash D(Z) \cup \Theta^* \cup \cup_{\gamma \in \Gamma} {\Theta_\gamma}^* \supset \epsilon^*$$

where

$$Z = Var(\Gamma^*) - Var(\Theta^* \cup (\cup_{\gamma \in \Gamma} \Theta_\gamma^*) \supset \epsilon^*).$$

Since $Z \cap X = \emptyset$ by definition of Z, it is sufficient to show that

$$CL_v(sp) \vdash D(X) \cup D(Z) \cup \Theta \cup (\cup_{\gamma \in \Gamma} \Theta_\gamma) \supset \epsilon.$$

By inductive hypothesis, $CL_v(sp) \vdash D(X) \cup \Theta \cup \Gamma \supset \epsilon$ and $CL_v(sp) \vdash D(X) \cup \Theta_\gamma \supset \gamma$ for all $\gamma \in \Gamma$. Thus

$$CL_v(sp) \vdash D(X) \cup D(Y) \cup \Theta \cup (\cup_{\gamma \in \Gamma} \Theta_\gamma) \supset \epsilon,$$

because of rule 5 of $CL_v(sp)$, where Y is the set

$$Var(\Gamma) - Var(\Theta \cup (\cup_{\gamma\in\Gamma}\Theta_\gamma) \supset \epsilon).$$

Moreover $Y \cup X = Z \cup X$ by definition of Y and Z and hence the thesis follows.

Instantiation/Abstraction Assume that $CL_g(sp_X) \vdash \Delta^* \supset \epsilon^*$.

Then for all families Z of variables (for $CL_g(sp)$) and all $\Gamma_{Def} = \{D(t_z) \mid z \in Z\}$, $CL_g(sp_X) \vdash \phi$, where ϕ is

$$\Gamma_{Def} \cup \{\delta[t_z/z \mid z \in Z] \mid \delta \in \Delta^*\} \supset \epsilon^*[t_z/z \mid z \in Z].$$

For all t_z there exists one term t'_z s.t. $t'^*_z{=}t_z$. Thus $(\eta^*[t_z/z \mid z \in Z])^* = (\eta[t'_z/z \mid z \in Z])^*$ for all elementary formulas η and hence ϕ is also

$$(\Gamma'_{Def})^* \cup (\{\delta[t'_z/z \mid z \in Z] \mid \delta \in \Delta\})^* \supset (\epsilon[t'_z/z \mid z \in Z])^*,$$

where Γ'_{Def} is $\{D(t'_z) \mid z \in Z\}$. By inductive hypothesis, $CL_v(sp) \vdash D(X) \cup \Delta \supset \epsilon$; thus, because of rule 6 of $CL_v(sp)$, $CL_v(sp) \vdash \phi'$, where ϕ' is

$$\Gamma'_{Def} \cup \{\delta[t'_z/z \mid z \in Z] \mid \delta \in \Delta \cup D(X)\} \supset \epsilon[t'_z/z \mid z \in Z].$$

Finally, since $X \cap Z = \emptyset$, because of the assumption that all variables in X do not appear in any proof of $CL_g(sp)$, $D(x)[t_z/z \mid z \in Z] = D(x)$ for all $x \in X$, and hence

$$\{\delta[t_z/z \mid z \in Z] \mid \delta \in \Delta \cup D(X)\} = D(X) \cup \{\delta[t_z/z \mid z \in Z] \mid \delta \in \Delta\}$$

so that ϕ' is

$$D(X) \cup \Gamma'_{Def} \cup \{\delta[t'_z/z \mid z \in Z] \mid \delta \in \Delta\} \supset \epsilon[t'_z/z \mid z \in Z]$$

and hence the thesis follows.

Elimination Assume that $CL_g(sp_X) \vdash \Delta^*_j \cup \Gamma^*_j \supset \epsilon^*$ for all $j \in J$ and that for all $\Psi \in \mathsf{Sec}(\{\Gamma^*_j \mid j \in J\})$ there exist $t, t' \in T_\Sigma(X)$ s.t. $D(t), D(t')$ and $t{=}t'$ belong to Ψ. Then, by inductive hypothesis, $CL_v(sp) \vdash D(X) \cup \Delta_j \cup \Gamma_j \supset \epsilon$ for all $j \in J$ and, by definition of Γ^*_j, for all $\Psi \in \mathsf{Sec}(\{\Gamma_j \mid j \in J\})$ there exist $t, t' \in T_\Sigma(X)$ s.t. $D(t), D(t')$ and $t{=}t'$ belong to Ψ. Thus, by elimination rule of $CL_v(sp)$, $CL_v(sp) \vdash D(X) \cup (\cup_{j\in J}\Delta_j) \supset \epsilon$ follows, too. $\square$

Prop. 4.4.14 If $CL_g(sp)$ is gseq-complete for all conditional specifications sp, then $CL_v(sp')$ is seq-complete for all families of variables and all conditional specifications sp'.

Proof. Let $\Sigma = (S, F)$ be a signature, $sp = (\Sigma, Ax)$ be a conditional specification and X be an S-sorted family of variables. Using the notation of Lemma 4.4.13, it is shown that the gseq-completeness of $CL_g(sp_X)$ implies the seq-completeness of $CL_v(sp)$ for X and sp.

Assume that $CL_g(sp_X)$ is gseq-complete, ϵ is an elementary formula over $\Sigma = (S, F)$ and X s.t. $CL_v(sp) \not\vdash D(X') \supset \epsilon$ for every $X' \subseteq X$ and show that there exists a model A s.t. $A \not\models_{\mathcal{PAR}} D(X) \supset \epsilon$, i.e. that there exists a valuation $V: X \to A$ s.t. $A \not\models_{\mathcal{PAR}_V} \epsilon$.

To this end it is sufficient to prove that there exists a model B of sp_X s.t. $B \not\models_{\mathcal{PAR}} \epsilon^*$; indeed the thesis follows for A defined by $s^A = s^B$ for all $s \in S$, $op^A = op^B$ for all $op \in F$ (i.e. A is the Σ-reduct of B) and V defined by $V(x) = {op_x}^B$, which is well defined because of the axioms $D(op_x)$. In order to prove that there exists such a model B, $CL_g(sp_X)$ being gseq-complete, it is sufficient to show that $CL_g(sp_X) \not\vdash \epsilon^*$. Because of Lemma 4.4.13, if $CL_g(sp_X) \vdash \epsilon^*$, then $CL_v(sp) \vdash D(X) \supset \epsilon$ and hence $CL_g(sp_X) \not\vdash \epsilon^*$, because $CL_v(sp) \not\vdash D(X) \supset \epsilon$. $\square$

Notation. Since in this section the focus is mostly on ground formulas and the system $CL_g(sp)$, some short notations is introduced first : in the sequel $\mathrm{EEq}(CL_g(sp), \emptyset)$ will be denoted by $\mathrm{EEq}(sp)$, $\mathrm{NF}(CL_g(sp), \emptyset)$ by $\mathrm{NF}(CL_g(sp))$ and $\mathsf{Fr}(CL_g(sp), \emptyset)$ by $I(sp)$. $\square$

In the literature, in the case both of partial positive conditional and of total conditional specifications, any system $L(sp)$ for sp is shown to be geeq-complete by proving that $I = T_\Sigma / \equiv^L (sp)$ is a model, since $I \models \epsilon$ iff $L(sp) \vdash \epsilon$ for all $\epsilon \in \mathrm{EEq}(sp, \emptyset)$. Unfortunately the same proof technique cannot be adopted in the case of (possibly) non-positive conditional specifications, since in general they do not have an initial model and if $I \in PMod(sp)$, then it is initial in $PMod(sp)$ (see e.g. Theorem 2.4.17). In other words it cannot be given a model which does not satisfy *all* undeducible ground equalities, but for each undeducible ground equality ϵ a model which does not satisfy ϵ has to be exhibited. To do this, first only those $\epsilon \in \mathrm{EEq}(sp)$ s.t. $CL_g(sp) \not\vdash \epsilon$ are considered and a model A_ϵ of sp is built s.t. $A_\epsilon \not\models_{\mathcal{PAR}} \epsilon$; in particular an enrichment sp_ϵ of sp is exhibited s.t. $CL_g(sp_\epsilon) \not\vdash \epsilon$ and $I(sp_\epsilon)$ is a model of sp_ϵ, so that $A_\epsilon = I(sp_\epsilon) \not\models_{\mathcal{PAR}} \epsilon$ and obviously $A_\epsilon \in PMod(sp)$, because sp_ϵ is an enrichment of sp.

Then for every $\eta \notin \mathrm{EEq}(sp)$ s.t. $CL_g(sp) \not\vdash \eta$ an enrichment sp_η of sp s.t. $\eta \in \mathrm{EEq}(sp_\eta)$ and $CL_g(sp_\eta) \not\vdash \eta$ is exhibited, so that the problem reduces to the above case.

Consider now the first problem, the building of such a sp_ϵ. As shown in Sect. 2.4, $I(sp)$ is not a model iff a naughty formula exists, i.e. a conditional sentence that is an instantiation of an axiom of sp by defined terms (snf_1) and s.t. its premises hold in $I(sp)$ (snf_2) while its consequence does not (snf_3). Thus the intuitive idea is to try to clear out the set $\mathrm{NF}(CL_{\mathrm{g}}(sp))$ of naughty formulas, by adding as axioms of the enrichment suitable elementary formulas making false snf_2 or snf_3 in the enriched specification for all formulas in $\mathrm{NF}(CL_{\mathrm{g}}(sp))$ (the set of these formulas is called a "resolving choice"). Obviously it is possible that in the enriched specification a conditional formula is naughty, which in sp was not, because it was not an instantiation of an axiom by *defined* terms, or because one of its premises was not deducible; so that also the enriched specification has not an initial model. Therefore a wider class of formulas has to be considered, the "possibly naughty" formulas.

Def. 4.4.15

- For a given conditional specification sp, the set $\mathrm{PNF}(sp)$ (for Possibly Naughty Formulas) consists of all ground conditional formulas ϕ s.t.
  - $CL_g(sp) \vdash \phi$;
  - $CL_g(sp) \not\vdash cons(\phi)$.
- An *r-choice* (for resolving choice) C is a set of ground elementary formulas s.t. for all $\phi \in \mathrm{PNF}(sp)$
  $(prem(\phi) \cap \mathrm{EEq}(sp)) \subseteq C$ implies that if $cons(\phi) \notin C$, then $(t{=}t') \in prem(\phi) - \mathrm{EEq}(sp)$ exists s.t. both conditions:
  - $(t{=}t'), (t'{=}t) \notin C$ and
  - one between $D(t)$ or $D(t')$ belongs to C

  hold.
- The set of all r-choices is denoted by *R-Choice*. □

The first intermediate result states that the resolving choices are really resolving, i.e. that the enriched specifications have initial model.

Lemma 4.4.16 For all conditional specifications $sp = (\Sigma, Ax)$ and all r-choices C, $\mathrm{NF}(\Sigma, \mathrm{Ax} \cup \mathrm{C}) = \emptyset$.
Proof. Let C be an r-choice for sp and sp' be the conditional specification $(\Sigma, Ax{\cup}C)$; in order to show that $\mathrm{NF}(sp')$ is empty assume that the conditions nf_1

and nf_2 hold for a conditional formula ϕ and the system $CL_g(sp')$ and show that the condition nf_3 does not hold. In particular it is shown that $CL_g(sp) \vdash cons(\phi)$ or $cons(\phi)$ belongs to C.

From nf_1, because of rules 5 and 6, $CL_g(sp') \vdash \phi$ and hence, by the deduction theorem, there exists a subset Γ of C s.t. $CL_g(sp) \vdash \Gamma \cup prem(\phi) \supset cons(\phi)$. Moreover, because of nf_2, $I(sp') \models \delta$ for all $\delta \in prem(\phi)$ and hence, by definition of $I(sp')$, $CL_g(sp') \vdash \delta$ for all $\delta \in prem(\phi) \cap \mathrm{EEq}(sp')$. Thus, by the deduction theorem, for all $\delta \in prem(\phi) \cap \mathrm{EEq}(sp')$ a subset Γ_δ of C exists s.t. $CL_g(sp) \vdash \Gamma_\delta \supset \delta$.

Thus, because of rule 5, $CL_g(sp) \vdash \phi'$, where ϕ' is the conditional formula

$$(\Gamma \cup \cup_{\delta \in prem(\phi) \cap \mathrm{EEq}(sp')} \Gamma_\delta) \cup (prem(\phi) - \mathrm{EEq}(sp')) \supset cons(\phi).$$

Therefore either $CL_g(sp) \vdash cons(\phi)$ or $\phi' \in \mathrm{PNF}(sp)$. Assume that $\phi' \in \mathrm{PNF}(sp)$ and show that $cons(\phi) \in C$. By definition of sp and sp', $\mathrm{EEq}(sp) \subseteq \mathrm{EEq}(sp')$ and hence $prem(\phi') \cap \mathrm{EEq}(sp) \subseteq prem(\phi') \cap \mathrm{EEq}(sp')$; moreover, by definition of ϕ',

$$prem(\phi') \cap \mathrm{EEq}(sp') \subseteq (\Gamma \cup \cup_{\delta \in prem(\phi) \cap \mathrm{EEq}(sp')} \Gamma_\delta) \subseteq C,$$

so that $prem(\phi') \cap \mathrm{EEq}(sp) \subseteq C$. Thus, by definition of r-choice, either $cons(\phi') \in C$, or there exists $(t{=}t') \in prem(\phi') - \mathrm{EEq}(sp)$ s.t. $(t{=}t'), (t'{=}t) \notin C$ and $D(t)$ or $D(t')$ belongs to C.

Finally, by definition of ϕ', for every $(t{=}t') \in (prem(\phi') - \mathrm{EEq}(sp))$ either $(t{=}t') \in (\Gamma \cup \cup_{\delta \in prem(\phi) \cap \mathrm{EEq}(sp')} \Gamma_\delta)$ and hence $(t{=}t') \in C$, or $(t{=}t') \in (prem(\phi) - \mathrm{EEq}(sp'))$ and hence, by definition of $\mathrm{EEq}(sp')$, neither $D(t) \in C$ nor $D(t') \in C$. Therefore, by definition of choice, $cons(\phi') \in C$. $\square$

Thus for all r-choices C, the enriched specification has an initial model; now it is shown that for every $\epsilon \in \mathrm{EEq}(sp)$ s.t. $CL_g(sp) \nvdash \epsilon$ there exists at least one r-choice s.t. also in the enriched specification ϵ does not hold.

Lemma 4.4.17 If $sp = (\Sigma, Ax)$ is a conditional specification and ϵ is an elementary ground formula s.t. $CL_g(sp) \nvdash \epsilon$, then there exists an r-choice C s.t. $CL_g(\Sigma, Ax \cup C) \nvdash \epsilon$.

Proof. Assume by contradiction that $CL_g(\Sigma, Ax \cup C) \vdash \epsilon$ for all r-choices C and show that there exists a set Θ of conditional formulas s.t.

1. $cons(\theta) = \epsilon$ for all $\theta \in \Theta$;

2. $CL_g(sp) \vdash \theta$ for all $\theta \in \Theta$;

3. for every $\Psi \in \mathsf{Sec}(\{prem(\theta) \mid \theta \in \Theta\})$ there exist $t, t' \in T_{\Sigma|s}$ s.t. $D(t)$, $D(t')$ and $t{=}t'$ belong to Ψ;

thus, because of the elimination rule, $CL_g(sp) \vdash \epsilon$, contrary to the hypothesis. Because of the assumption that $CL_g(\Sigma, Ax \cup C) \vdash \epsilon$ for all r-choices C and of the deduction theorem, for each r-choice C there exists $\Gamma_C \subseteq C$ s.t. $CL_g(sp) \vdash \Gamma_C \supset \epsilon$; let Θ be the set $\cup_{C \in R-Choice} R(\Gamma_C \supset \epsilon)$, where $R(\Gamma_C \supset \epsilon)$ consists of the formulas $\Gamma_1 \cup \cup_{\gamma \in \Gamma_2} \Delta_\gamma \supset \epsilon$ s.t.:

- $\Gamma_1 \cup \Gamma_2 = \Gamma_C$;
- $\Gamma_1 \cap \Gamma_2 = \emptyset$
- $(\Delta_\gamma \supset \gamma) \in \mathrm{PNF}(sp)$ for all $\gamma \in \Gamma_2$.

Since $CL_g(sp) \vdash \psi$ for all $\psi \in \mathrm{PNF}(sp)$ and $CL_g(sp) \vdash \Gamma_C \supset \epsilon$ for all r-choices C, then, by rule 5, $CL_g(sp) \vdash \theta$ for all $\theta \in \Theta$; thus Θ satisfies conditions 1 and 2 and hence it is sufficient to show that Θ satisfies also condition 3.

Again the proof is by contradiction: assume that condition 3 does not hold, i.e. that there exists $\Psi \in \mathsf{Sec}(\{prem(\theta) \mid \theta \in \Theta\})$ s.t. for all $t, t' \in T_{\Sigma|s}$ at least one among $D(t)$, $D(t')$ and $t{=}t'$ does not belong to Ψ, and show that there exists $\theta \in \Theta$ s.t. $prem(\theta) \cap \Psi = \emptyset$, contrary to the assumption that $\Psi \in \mathsf{Sec}(\{prem(\theta) \mid \theta \in \Theta\})$.

In order to prove that there exists such a θ an r-choice C is built s.t. for all $\gamma \in C$ either $\gamma \notin \Psi$, or there exists $(\Delta_\gamma \supset \gamma) \in \mathrm{PNF}(sp)$ s.t. $\Delta_\gamma \cap \Psi = \emptyset$; thus $\theta' = [(\Gamma_C - \Psi) \cup \cup_{\gamma \in \Gamma_C \cap \Psi} \Delta_\gamma] \supset \epsilon$ belongs to $R(\Gamma_C \supset \epsilon)$ and hence to Θ, while $prem(\theta') \cap \Psi = \emptyset$.

Let C be $C_C \cup C_P$, for $C_C = \{\gamma \mid (\Delta \supset \gamma) \in \mathrm{PNF}(sp), \Delta \cap \Psi = \emptyset\}$ and $C_P = \{D(t) \mid t \in T_\Sigma, D(t) \notin \Psi\}$.

Thus for all $\gamma \in C$ either $\gamma \notin \Psi$, or there exists $(\Delta_\gamma \supset \gamma) \in \mathrm{PNF}(sp)$ s.t. $\Delta_\gamma \cap \Psi = \emptyset$ and hence it is sufficient to show that C is a choice. Let χ belong to $\mathrm{PNF}(sp)$ s.t. $prem(\chi) \cap \mathrm{EEq}(sp) \subseteq C$; then, by definition of choice, it is sufficient to show that $cons(\chi) \in C$, or $\star$ [there exists $(t{=}t') \in prem(\chi) - \mathrm{EEq}(sp)$ s.t. $(t{=}t') \notin C$ and $D(t)$ or $D(t')$ belongs to C]. Thus assume that $\star$ does not hold, i.e. that for all $(t{=}t') \in prem(\chi) - \mathrm{EEq}(sp)$ $(t{=}t') \in C$, or both $D(t)$ and $D(t')$ do not belong to C, and show that there exists $\chi' \in \mathrm{PNF}(sp)$ s.t. $prem(\chi') \cap \Psi = \emptyset$ and $cons(\chi') = cons(\chi)$, so that $cons(\chi') \in C_C$ and hence $cons(\chi) \in C$. Since $CL_g(sp) \vdash \psi$ for all $\psi \in \mathrm{PNF}(sp)$, $R(\psi) \subseteq \mathrm{PNF}(sp)$ for all $\psi \in \mathrm{PNF}(sp)$; so that in order to built such a χ' it is sufficient to show that for all $\eta \in prem(\chi) \cap \Psi$ there exists $(\Delta_\eta \supset \eta) \in \mathrm{PNF}(sp)$ s.t. $\Delta_\eta \cap \Psi = \emptyset$ and the thesis follows for $\chi' = [(prem(\chi) - \Psi) \cup (\cup_{\eta \in (prem(\chi) \cap \Psi)} \Delta_\eta)] \supset cons(\chi)$.

Let $\eta \in prem(\chi) \cap \Psi$; show that $\eta \in C$ and hence $\eta \in C \cap \Psi = C_C$ so that, by definition of C, there exists $(\Delta_\eta \supset \eta) \in \mathrm{PNF}(sp)$ s.t. $\Delta_\eta \cap \Psi = \emptyset$ If $\eta \in \mathrm{EEq}(sp)$, then, because of the assumption $prem(\chi) \cap \mathrm{EEq}(sp) \subseteq C$, $\eta \in C$. Otherwise $\eta \in prem(\chi) - \mathrm{EEq}(sp)$ has the form $t{=}t'$ and, because of the absurd hypothesis on Ψ and $t{=}t' \in \Psi$, $D(t)$ or $D(t')$ does not belong to Ψ, so that, because of definition of C_P, $D(t)$ or $D(t')$ belongs to C; thus, since $(t{=}t') \in C$, or both $D(t)$ and $D(t')$ do not belong to C, $\eta = (t{=}t') \in C$. □

Finally it is possible to show the gseq-completeness of $CL_g(sp)$.

Theorem 4.4.18 The system $CL_g(sp)$ is gseq-complete w.r.t. sp.
Proof. Let ϵ be an elementary ground formula, assume that $CL_g(sp) \not\vdash \epsilon$ and show that there exists a model A of $sp = (\Sigma, Ax)$ s.t. $A \not\models_{\mathcal{PAR}} \epsilon$. The proof is divided in two cases.

1. Let ϵ belong to $\mathrm{EEq}(sp)$.
   - If $\mathrm{NF}(CL_g(sp))$ is empty, then $A = T_\Sigma/\equiv^{CL_g(sp)}$ is a model, because of the Theorem 2.4.17, and, by construction of $\equiv^{CL_g(sp)}$, $A \not\models_{\mathcal{PAR}} \epsilon$.
   - Otherwise an r-choice C exists s.t. $CL_g(\Sigma, Ax \cup C) \not\vdash \epsilon$, by Lemma 4.4.17. Moreover $NF(CL_g(sp'))$ is empty, where $sp' = (\Sigma, Ax \cup C)$, because of Lemma 4.4.16; thus $A = T_\Sigma/\equiv^{CL_g(sp')}$ is a model of sp', because of Theorem 2.4.17. Finally A belongs to $PMod(sp)$, since $PMod(sp') \subseteq PMod(sp)$ by definition of sp', and moreover $A \not\models_{\mathcal{PAR}} \epsilon$, by definition of A.

2. Let ϵ have the form $t{=}t'$, $CL_g(sp) \not\vdash D(t)$, and $CL_g(sp) \not\vdash D(t')$; if there exists a conditional specification sp' s.t.

   (a) $PMod(sp') \subseteq PMod(sp)$;

   (b) $\epsilon \in \mathrm{EEq}(CL_g(sp'))$;

   (c) $CL_g(sp') \not\vdash \epsilon$;

   then, because of 1 and of conditions 2b and 2c, there exists a model A of sp' s.t $A \not\models_{\mathcal{PAR}} \epsilon$ and hence, because of 2a, A is also a model of sp which does not satisfy ϵ. Therefore it is sufficient to show that there exists such a sp'.

   Let sp_1 be the specification $(\Sigma, Ax \cup \{D(t)\})$ and sp_2 be the specification $(\Sigma, Ax \cup \{D(t')\})$; it is shown that $CL_g(sp_1) \not\vdash t{=}t'$ or $CL_g(sp_2) \not\vdash t{=}t'$. By contradiction assume that $CL_g(sp_1) \vdash t{=}t'$ and $CL_g(sp_2) \vdash t{=}t'$ and prove that $CL_g(sp) \vdash t{=}t'$.

Because of the absurd hypothesis and of the deduction theorem, both $CL_g(sp) \vdash D(t) \supset t{=}t'$ and $CL_g(sp) \vdash D(t') \supset t{=}t'$; moreover, by rule 2.2b, $CL_g(sp) \vdash t{=}t' \supset t'{=}t$ and $CL_g(sp) \vdash t'{=}t \supset t{=}t'$ and hence, by rule 5, $CL_g(sp) \vdash t{=}t' \supset t{=}t'$. Thus, applying the elimination rule to the set $\{D(t) \supset t{=}t', D(t') \supset t{=}t', t{=}t' \supset t{=}t'\}$, $CL_g(sp) \vdash t{=}t'$. Therefore $CL_g(sp_1) \not\vdash t{=}t'$, and in this case let sp' be sp_1, or $CL_g(sp_2) \not\vdash t{=}t'$, and in this case let sp' be sp_2. In any case sp' satisfies conditions 2a, 2b, 2c by definition. □

Theorem 4.4.19 Let sp be the conditional specification (Σ, Ax) and X be a family of variables; the conditional system $CL_v(sp)$ is seq-complete for X and sp.
Proof. From Prop. 4.4.14, $CL_g(sp')$ being gseq-complete for all conditional specifications sp', because of Theorem 4.4.18. □

Finitary Deduction

If all the axioms of a conditional specification sp have a finite set of premises, then the system $CL_v(sp)$ can be specialized to obtain an seq-complete system whose rules have a finitary number of premises and only deal with finitary formulas. This is in particular the case for every finitary higher-order specification, because the axioms introduced by the skolemization procedure are finitary.

Def. 4.4.20 Let $sp = (\Sigma, Ax)$ be a conditional specification s.t. Δ is finite for all $\Delta \supset \epsilon$ in Ax. Let $CL_f(sp)$ be the inference system consisting of the axioms in Ax, of the axioms 1…4 of $CL_v(sp)$ (definedness of variables, congruence, strictness, definedness and equality) and of the following inference rules; where any formula is assumed to be finitary:

5_f *Modus Ponens*

$$\frac{\Delta \cup \{\gamma\} \supset \epsilon, \Delta_\gamma \supset \gamma}{D(Var(\gamma) - Var(\Delta \cup \Delta_\gamma \supset \epsilon)) \cup (\Delta \cup \Delta_\gamma) \supset \epsilon}$$

6_f *Instantiation/Abstraction*

$$\frac{\Delta \supset \epsilon}{\{D(t)\} \cup \{\delta[t/x] \mid \delta \in \Delta\} \supset \epsilon[t/x]}$$

where $t \in T_\Sigma(Var)_{|s}, x \in X_s$.

7_f *Elimination*

$$\frac{(\Delta_1 \cup \{D(t)\}) \supset \epsilon, (\Delta_2 \cup \{D(t')\}) \supset \epsilon, (\Delta_3 \cup \{t{=}t'\}) \supset \epsilon}{D(Var(t{=}t')) \cup (\Delta_1 \cup \Delta_2 \cup \Delta_3) \supset \epsilon}$$

The seq-completeness of $CL_f(sp)$ follows from the one of $CL_v(sp)$, because $CL_v(sp) \vdash \Delta \supset \epsilon$ implies that there exists a finite $\Gamma \subseteq \Delta$ s.t. $CL_f(sp) \vdash \Gamma \supset \epsilon$. To prove this claim two intermediate results are needed. The first lemma is stated in a merely combinatorial form and guarantees that every application of the elimination rule to a possibly infinite set of finitary premises may be replaced by an application of the elimination rule to a *finite* set of finitary premises. The second lemma states that an application of the elimination rule to a *finite* set of finitary premises can be replaced by a *finite* sequence of application of the rule of finitary elimination 9_f.

Notation. Let Val be a denumerable set of values, $\mathcal{R} \subseteq \wp_{Fin}(\mathsf{Val})$ be a relation over Val and Γ be a (possibly more than denumerable) collection of subsets of Val. It is denoted by $\mathsf{Sec}(\Gamma)$ the collection of all possible *sections* of Γ, i.e. of all subsets σ of Val s.t. $\sigma = \{v_\gamma \mid \gamma \in \Gamma\}$ for some $v_\gamma \in \gamma$ for each $\gamma \in \Gamma$, and say that Γ is $\mathcal{R}$-*ininfluent* iff for every section σ of Γ there exists $R \in \mathcal{R}$ s.t. $R \subseteq \sigma$. $\square$

Lemma 4.4.21 Let Val be a denumerable set of values, $\mathcal{R} \subseteq \wp_{Fin}(\mathsf{Val})$ be a relation over Val and Γ be a (possibly more than denumerable) collection of subsets of Val. If Γ is an $\mathcal{R}$-ininfluent collection of subsets of Val s.t. γ is finite for all $\gamma \in \Gamma$, then there exists an $\mathcal{R}$-ininfluent finite subset of Γ.

Proof. If Γ is finite, then the thesis is trivial; thus assume that Γ is infinite. Since Val is denumerable and Γ is a collection of finite subsets of Val, Γ is denumerable, too. Fix an enumeration for Γ and denote $\Gamma = \{\Gamma_i \mid i \in \mathbb{N}\}$. Now a tree is built whose finite paths are all and only the sections of $\{\Gamma_i \mid i \leq n\}$ for all $n \in \mathbb{N}$ which do not contain any element of $\mathcal{R}$ and this tree is shown to be finite. The tree is inductively defined as follows.

T_0 is just the root, labeled by the empty set.

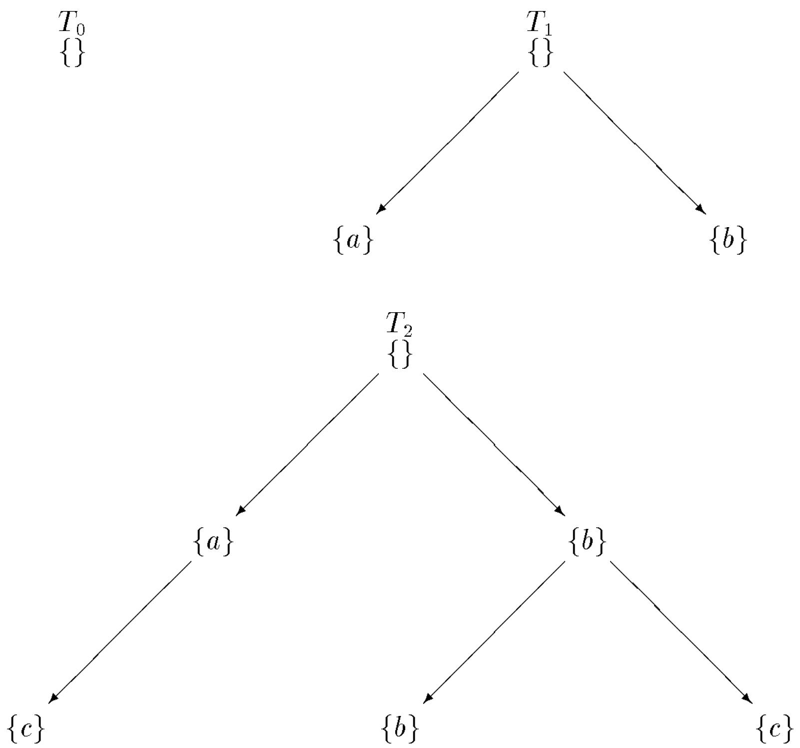
T_{n+1} is the tree obtained from T_n by the following rule:

> for every leaf l of T_n let $path(l)$ denote the set of the labels of the nodes from the root to l; then to each leaf l of T_n at depth n a son labeled $\{\gamma\}$ is added for all $\gamma \in \Gamma_n$ s.t. $path(l) \cup \{\gamma\}$ does not contain elements of $\mathcal{R}$
>
> (Note that in this way if at step k a son has not be added to some leaf, then at any further step, say n, this leaf is at depth $k < n$ and hence no sons are added any more).

Assume by contradiction that there does not exist $n \in \mathbb{N}$ s.t. $T_n = T_{n+1}$. Then by construction an infinite tree is obtained, which is finitely branching, because every Γ_n is finite; hence, by the König lemma, there exists an infinite path. But an infinite path, by construction, should be a section for Γ which does not contain any element of $\mathcal{R}$, contrary to the assumption that Γ is $\mathcal{R}$-ininfluent. Therefore there exists $\bar{n} \in \mathbb{N}$ s.t. $T_{\bar{n}} = T_{\bar{n}+1}$. Finally $\Gamma' = \{\Gamma_0 \dots \Gamma_{\bar{n}+1}\}$ is $\mathcal{R}$- ininfluent. Indeed let $\{\gamma_0 \dots \gamma_{\bar{n}+1}\}$ be a section for Γ'; then either there exists $R \in \mathcal{R}$ s.t. $R \subseteq \{\gamma_0 \dots \gamma_{\bar{n}}\}$ or $\{\gamma_0 \dots \gamma_{\bar{n}}\}$ is $path(l)$ for some leaf l of $T_{\bar{n}}$ and hence for each $\gamma \in \Gamma_{\bar{n}+1}$ there exists $R \in \mathcal{R}$ s.t. $R \subseteq \{\gamma_0 \dots \gamma_{\bar{n}}, \gamma\}$, because $T_{\bar{n}} = T_{\bar{n}+1}$, so that in particular $\{\gamma_0 \dots \gamma_{\bar{n}+1}\}$ contains an element of $\mathcal{R}$. Thus in both cases there exists $R \in \mathcal{R}$ s.t. $R \subseteq \{\gamma_0 \dots \gamma_{\bar{n}+1}\}$ and hence Γ' is $\mathcal{R}$-ininfluent. $\square$

Just to get the intuition of the construction of such trees T_n, consider the following simple example.

- $\mathsf{Val} = \{a, b, c, d, e, \dots\}$;
- $\mathcal{R} = \{\{a, b\}, \{a, c, d\}, \{b, d\}\}$;
- $\Gamma_0 = \{a, b\}$; $\Gamma_1 = \{b, c\}$; $\Gamma_2 = \{a, d, e\}$;...



$$\begin{array}{c} T_3 \\ \{\} \end{array}$$

$$\begin{tikzcd}
 & & \{\} \arrow[dl] \arrow[dr] & & \\
 & \{a\} \arrow[dl] & & \{b\} \arrow[dl] \arrow[dr] & \\
\{c\} \arrow[d] \arrow[dr] & & \{b\} \arrow[dr] & & \{c\} \arrow[d] \\
\{c\} & \{e\} & & \{e\} & \{e\}
\end{tikzcd}$$

In order to show that an application of the elimination rule to a *finite* set of finitary premises can be replaced by a *finite* sequence of applications of the rule of finitary elimination 7_f, a preliminary result is needed first.

Lemma 4.4.22 Let $\Theta_j \cup \Gamma_j \supset \epsilon$ be conditional formulas for $j = 1 \dots k$; if the following conditions are satisfied, then $CL_f(sp) \vdash D(Y) \cup \Theta \supset \epsilon$ holds for some finite $\Theta \subseteq \cup_{j=1\dots k}\Theta_j$ and $Y \subseteq Var(\cup_{j=1\dots k}\Gamma_j)$.

1. $CL_f(sp) \vdash \Theta_j \cup \Gamma_j \supset \epsilon$ for $j = 1 \dots k$;
2. there exist terms t_i, t'_i for $i = 1 \dots n$ s.t. for each section σ of $\mathcal{R} = \{R(t_i, t'_i) \mid i = 1 \dots n\}$, where $R(t, t') = \{D(t), D(t'), t = t'\}$, there exists $j \in J$ s.t. $\Gamma_j \subseteq \sigma$.

Proof. The proof is done by induction over n.

- Let n be 1; then from condition 2 there exists $j \in \{1 \dots k\}$ s.t. $\Gamma_j = \emptyset$, and in this case $CL_f(sp) \vdash \Theta_j \supset \epsilon$, because of condition 1, so the thesis follows for $\Theta = \Theta_j$ and $Y = \emptyset$, or there are j_1, j_2, j_3 s.t. $\Gamma_{j_1} = \{D(t_1)\}$, $\Gamma_{j_2} = \{D(t'_1)\}$, $\Gamma_{j_3} = \{t_1{=}t'_1\}$ and in this case, because of rule 7_f, $CL_f(sp) \vdash D(Var(t_1{=}t'_1)) \cup (\Theta_{j_1} \cup \Theta_{j_2} \cup \Theta_{j_3}) \supset \epsilon$, so the thesis follows for $Y = Var(t_1{=}t'_1)$ and $\Theta = \Theta_{j_1} \cup \Theta_{j_2} \cup \Theta_{j_3}$.

- Assume that for all conditional formulas $\Theta'_j \cup \Gamma'_j \supset \epsilon'$ for $j = 1 \dots k'$ satisfying conditions 1 and 2 (for $i = 1 \dots n$) there exist $\Theta' \subseteq \cup_{j=1\dots k'} \Theta'_j$ and $Y' \subseteq Var(\cup_{j=1\dots k'} \Gamma'_j)$ s.t. $CL_f(sp) \vdash D(Y') \cup \Theta' \supset \epsilon'$ and that $\{\Theta_j \cup \Gamma_j \supset \epsilon \mid j = 1 \dots k\}$ satisfies conditions 1 and 2 for $n+1$.

  Then for every section σ of $\mathcal{R} = \{R(t_i, t'_i) \mid i = 1 \dots n\}$ the sets Δ_σ and Ψ_σ are defined as follows. Let σ_1 be $\sigma \cup \{D(t_{n+1})\}$, σ_2 be $\sigma \cup \{D(t'_{n+1})\}$ and σ_3 be $\sigma \cup \{t_{n+1}{=}t'_{n+1}\}$; because of condition 2, there exist $j_1, j_2, j_3 \in \{1 \dots k\}$ s.t. $\Gamma_{j_i} \subseteq \sigma_i$ for $i = 1, 2, 3$. If there exists $i \in \{1, 2, 3\}$ s.t. $\Gamma_{j_i} \subseteq \sigma$, then define $\Psi_\sigma = \Theta_{j_i}$ and $\Delta_\sigma = \Gamma_{j_i}$, so that $CL_f(sp) \vdash \Delta_\sigma \cup \Psi_\sigma \supset \epsilon$ from condition 1 and $\Delta_\sigma \subseteq \sigma$ by construction. Otherwise $\Gamma_{j_1} = \Gamma'_{j_1} \cup \{D(t_{n+1})\}$, $\Gamma_{j_2} = \Gamma'_{j_2} \cup \{D(t'_{n+1})\}$ and $\Gamma_{j_3} = \Gamma'_{j_3} \cup \{t_{n+1}{=}t'_{n+1}\}$; in this case define $\Psi_\sigma = \Theta_{j_1} \cup \Theta_{j_2} \cup \Theta_{j_3} \cup D(Var(t_{n+1} = t'_{n+1}))$ and $\Delta_\sigma = \Gamma'_{j_1} \cup \Gamma'_{j_2} \cup \Gamma'_{j_3}$; also in this case $CL_f(sp) \vdash \Delta_\sigma \cup \Psi_\sigma \supset \epsilon$, because of rule 7_f and condition 1, and $\Delta_\sigma \subseteq \sigma$ by construction.

  Thus the conditional formulas $\Delta_\sigma \cup \Psi_\sigma \supset \epsilon$ for all sections σ of $\mathcal{R}$ satisfy the condition 1 and condition 2 for $\mathcal{R} = \{R(t_i, t'_i) \mid i = 1 \dots n\}$. Therefore, because of the inductive hypothesis, there exist finite $\Theta' \subseteq \cup_{\sigma \in \mathsf{Sec}(\mathcal{R})} \Psi_\sigma$ and $Y' \subseteq Var(\cup_{\sigma \in \mathsf{Sec}(\mathcal{R})} \Delta_\sigma)$ s.t. $CL_f(sp) \vdash D(Y') \cup \Theta' \supset \epsilon$.

  Since both $Var(\cup_{\sigma \in \mathsf{Sec}(\mathcal{R})} \Delta_\sigma) \subseteq Var(\cup_{j=1\dots k} \Gamma_j)$ and $\cup_{\sigma \in \mathsf{Sec}(\mathcal{R})} \Psi_\sigma \subseteq \cup_{j=1\dots k} \Theta_j \cup Var(t_{n+1}{=}t'_{n+1})$, $CL_f(sp) \vdash D(Y) \cup \Theta \supset \epsilon$, where $\Theta = \Theta' - D(Var(t_{n+1} = t'_{n+1}))$ and $Y = Y' \cup (\Theta' - \Theta)$. □

Lemma 4.4.23 If $CL_f(sp) \vdash \Theta_j \cup \Gamma_j \supset \epsilon$ for all $j \in J$, $\Gamma = \{\Gamma_j \mid j \in J\}$ is ininfluent and J is finite, then $CL_f(sp) \vdash D(Y) \cup \Theta \supset \epsilon$ holds for some finite $\Theta \subseteq \cup_{j \in J} \Theta_j$ and $Y \subseteq Var(\cup_{j \in J} \Gamma_j)$.
Proof. By Prop. 4.4.9, $I = \{1 \dots k\}$ and terms t_i, t'_i for $i \in I$ exist s.t. for every section σ of $\mathcal{R} = \{R(t_i, t'_i) \mid i \in I\}$, where $R(t, t') = \{D(t), D(t'), t = t'\}$, there exists $j \in J$ s.t. $\Gamma_j \subseteq \sigma$. Thus Lemma 4.4.22 applies. □

Theorem 4.4.24 Let $sp = (\Sigma, Ax)$ be a conditional specification s.t. Δ is finite for all $\Delta \supset \epsilon$ in Ax and X be a family of variables. The system $CL_f(sp)$ is seq-complete for sp and X.
Proof. The proof relies over the seq-completeness of $CL_v(sp)$. Indeed it is shown by induction over the rules of $CL_v(sp)$, that $CL_v(sp) \vdash \Delta \supset \epsilon$ implies that a finite $\Gamma \subseteq \Delta$ exists s.t. $CL_f(sp) \vdash \Gamma \supset \epsilon$. Thus for every elementary formula ϵ if $A \models D(X) \supset \epsilon$ for all $A \in PMod(sp)$, then $CL_v(sp) \vdash D(X') \supset \epsilon$ for some $X' \subseteq X$, because of the seq-completeness of $CL_v(sp)$, and hence $CL_f(sp) \vdash D(Y) \supset \epsilon$ for a finite $Y \subseteq X' \subseteq X$. Consider the inductive proof.

The proper axioms and the axioms 1...4 are common to both the systems and have finitary premises, so that the thesis trivially follows.

Assume that the premises of rule 5 have been deduced, i.e. that $CL_v(sp) \vdash \Delta \cup \Gamma \supset \epsilon$ and $CL_v(sp) \vdash \Delta_\gamma \supset \gamma$ for every $\gamma \in \Gamma$, and show that $CL_f(sp) \vdash \Gamma'' \supset \epsilon$ for a finite

$$\Gamma'' \subseteq (\Delta \cup \cup_{\gamma \in \Gamma} \Delta_\gamma) \cup D(Var(\Gamma) - Var((\Delta \cup \cup_{\gamma \in \Gamma} \Delta_\gamma) \supset \epsilon).$$

Because of the inductive hypothesis, $CL_f(sp) \vdash \Delta' \cup \Gamma' \supset \epsilon$, for some finite $\Gamma' \subseteq \Gamma$ and $\Delta' \subseteq \Delta$, and $CL_f(sp) \vdash \Delta'_\gamma \supset \gamma$ for every $\gamma \in \Gamma$ and some finite $\Delta'_\gamma \subseteq \Delta_\gamma$. Let Γ' be $\{\gamma_1 \dots \gamma_n\}$; it is easy to check that applying n times the rule 5_f the γ_i can be replaced one at a time by Δ'_{γ_i} and then, always by rule 5_f, the superfluous variables (i.e. variables of γ_i already occurring in Δ_{γ_j}) can be removed so that

$$CL_f(sp) \vdash (\Delta' \cup \cup_{i=1\dots n} \Delta'_{\gamma_i}) \cup D(Var(\Gamma') - Var((\Delta' \cup \cup_{i=1\dots n} \Delta'_{\gamma_i}) \supset \epsilon) \supset \epsilon$$

and hence the thesis follows for

$$\Gamma'' = (\Delta' \cup \cup_{i=1\dots n} \Delta'_{\gamma_i}) \cup D(Var(\Gamma') - Var((\Delta' \cup \cup_{i=1\dots n} \Delta'_{\gamma_i}).$$

Assume that the premise of rule 6 has been deduced, i.e. that $CL_v(sp) \vdash \Delta \supset \epsilon$, and show that there exists a finite

$$\Gamma \subseteq \{D(t_x) \mid x \in X_s, s \in S\} \cup \{\delta[t_x/x \mid x \in X_s, s \in S] \mid \delta \in \Delta\}$$

s.t. $CL_f(sp) \vdash \Gamma \supset \epsilon$. Because of the inductive hypothesis, $CL_f(sp) \vdash \Gamma' \supset \epsilon$ for some finite $\Gamma' \subseteq \Delta$. It is easy to check that applying n times the rule 5_f, starting from $\Gamma' \supset \epsilon$, the system $CL_f(sp)$ deduces

$$\{D(t_{x_1}) \dots D(t_{x_n})\} \cup \{\gamma[t_{x_1}/x_1, \dots, t_{x_n}/x_n] \mid \gamma \in \Gamma'\} \supset \epsilon[t_{x_1}/x_1 \dots t_{x_n}/x_n],$$

for $\{x_1 \dots x_n\}$ the variables of X appearing in $\Gamma' \supset \epsilon$.

Thus

$$CL_f(sp) \vdash \{D(t_{x_1}) \dots D(t_{x_n})\} \cup \{\gamma[t_x/x \mid x \in X] \mid \gamma \in \Gamma'\} \supset \epsilon[t_x/x \mid x \in X]$$

and $\Gamma = \{D(t_{x_1}) \dots D(t_{x_n})\} \cup \{\gamma[t_x/x \mid x \in X] \mid \gamma \in \Gamma'\}$ is a finite subset of $\{D(t_x) \mid x \in X_s, s \in S\} \cup \{\delta[t_x/x \mid x \in X_s, s \in S] \mid \delta \in \Delta\}$, because $\{x_1 \dots x_n\} \subseteq X$, and $\Gamma' \subseteq \Delta$ is finite.

Thus the only non-trivial step is the elimination rule.

Assume that $CL_v(sp) \vdash \Theta_j \cup \Gamma_j \supset \epsilon$ for all $j \in J$, that $\Gamma = \{\Gamma_j \mid j \in J\}$ is ininfluent and show that $CL_f(sp) \vdash D(Y) \cup \Theta \supset \epsilon$ for some $\Theta \subseteq \cup_{j \in J} \Theta_j$ and

$Y \subseteq Var(\cup_{j\in J}\Gamma_j)$. Because of the inductive hypothesis, $CL_f(sp) \vdash \Theta'_j \cup \Gamma'_j \supset \epsilon$ follows from $CL_v(sp) \vdash \Theta_j \cup \Gamma_j \supset \epsilon$, where both $\Theta'_j \subseteq \Theta_j$ and $\Gamma'_j \subseteq \Gamma_j$ are finite, for all $j \in J$.

Moreover $\Gamma' = \{\Gamma'_j \mid j \in J\}$ is ininfluent, because $\Gamma'_j \subseteq \Gamma_j$ implies $\mathsf{Sec}(\Gamma') \subseteq \mathsf{Sec}(\Gamma)$. Thus, because of Lemma 4.4.21 for $\mathsf{Val} = \mathrm{EForm}(\Sigma, X)$ and $\mathcal{R} = \{\{D(t), D(t'), t{=}t'\} \mid t, t' \in T_\Sigma(Var)\}$, a finite $I \subseteq J$ exists s.t. $\{\Gamma'_i \mid i \in I\}$ is ininfluent and hence, because of Lemma 4.4.23, $CL_f(sp) \vdash D(Y) \cup \Theta \supset \epsilon$ for some $\Theta \subseteq \cup_{i\in I}\Theta'_i$ and $Y \subseteq Var(\cup_{i\in I}\Gamma'_i)$. As I and both Γ'_i and Θ'_i for every $i \in I$ are finite, also Θ and Y are finite; moreover $\Theta \subseteq \cup_{i\in I}\Theta'_i \subseteq \cup_{j\in J}\Theta'_j \subseteq \cup_{j\in J}\Theta_j$ and $Y \subseteq Var(\cup_{i\in I}\Gamma'_i) \subseteq Var(\cup_{j\in J}\Gamma_j)$ so that the thesis follows. □

Functional Deduction

Since the skolemization axioms are finitary, the translation along μ^E of the inference system $CL_f(sp)$ yields a sound and complete system for every finitary higher-order specification.

For the more general case of infinitary higher-order specifications, a sound and complete system can be obtained by translating along μ^E the system $CL_v(sp)$.

Def. 4.4.25 Let $\mathcal{L}_{\mathcal{PAR}}$ be the logic $(\mathbf{Sign}_{\mathcal{PAR}}, Sen_{\mathcal{PAR}}, Mod_{\mathcal{PAR}}, \models_{\mathcal{PAR}}, \vdash_{CL})$, where $\Gamma \vdash_{CL\Sigma} \phi$ iff $CL_v((\Sigma, \Gamma)) \vdash \phi$, and $\mathcal{L}_{\mathcal{PAR}_{Fin}}$ be the logic $(\mathbf{Sign}_{\mathcal{PAR}}, Sen_{\mathcal{PAR}_{Fin}}, Mod_{\mathcal{PAR}}, \models_{\mathcal{PAR}}, \vdash_{CLF})$, where $\Gamma \vdash_{CLF\Sigma} \phi$ iff $CL_f((\Sigma, \Gamma)) \vdash \phi$. □

It is immediate to check that $\vdash_{CL}$ and $\vdash_{CLF}$ are entailment systems.

Prop. 4.4.26 The logic $\mathcal{L}_{\mathcal{FPHO}} = (\mathbf{Sign}_{\mathcal{PHO}}, Sen_{\mathcal{FPHO}}, Mod_{\mathcal{PHO}}, \models_{\mathcal{PAR}}, \vdash_{CLF}{}^{\mu^E})$ is finitary equationally complete, i.e. for every either strong or existential equality ϵ on variables X and every set Γ of finitary conditional sentences if $\mathsf{A} \models_{\mathcal{PAR}} \gamma$ for all $\gamma \in \Gamma$ implies $\mathsf{A} \models_{\mathcal{PAR}} D(X) \supset \epsilon$, then $\Gamma \vdash_{CLF}{}^{\mu^E} D(X') \supset \epsilon$ for a finite $X' \subseteq X$.

The logic $\mathcal{L}_{\mathcal{PHO}} = (\mathbf{Sign}_{\mathcal{PHO}}, Sen_{\mathcal{PHO}}, Mod_{\mathcal{PHO}}, \models_{\mathcal{PAR}}, \vdash_{CL}{}^{\mu^E})$ is equationally complete, i.e. for every either strong or existential equality ϵ on variables X and every set Γ of finitary conditional sentences if $\mathsf{A} \models_{\mathcal{PAR}} \gamma$ for all $\gamma \in \Gamma$ implies $\mathsf{A} \models_{\mathcal{PAR}} D(X) \supset \epsilon$, then $\Gamma \vdash_{CL}{}^{\mu^E} D(X) \supset \epsilon$.

Proof. Because of Prop. 4.4.4, that applies because μ^E is logical simulation, i.e. a surjective map of institution. □

4.5 Building Proof Calculi

By applying the general result to the adjunction between entailment systems and proof calculi, an informative proof system is attached to any entailment system $\mathcal{E}$, provided a proof calculus $\mathcal{P}$ and a translation of $\mathcal{E}$ into the entailment system underlying $\mathcal{P}$ are given.

Prop. 4.5.1 Let $\underline{PCalc}$ denote the category of proof calculi with maps of proof calculi as arrows and $\underline{Ent}$ denote the category of entailment systems with maps of entailment systems as arrows.
Then the functor $(\_)^\sharp\colon \underline{Ent} \to \underline{PCalc}$, defined by

- $(\mathcal{E})^\sharp = (\mathbf{Sign}, Sen, \vdash, thm, Id_{\mathbf{Set}}, j)$, where thm is the functor sending each theory to the set of its theorems, described in Section 2.2 of [63], and j is the natural subfunctor inclusion $thm \subseteq Sen$ and
- $(\Phi, \alpha)^\sharp = (\Phi, \alpha, \alpha_{|thm})$, where $\alpha_{|thm}$ is the restriction of α to the theorem of the domain, for every map (Φ, α),

is the right adjoint of the forgetful functor $ent\colon \underline{PCalc} \to \underline{Ent}$, defined by $ent(\mathcal{P}) = (\mathbf{Sign}, Sen, \vdash)$ for every proof calculus $\mathcal{P} = (\mathbf{Sign}, Sen, \vdash, P, \mathrm{Pr}, \pi)$ and $ent(\Phi, \alpha, \gamma) = (\Phi, \alpha)$ for every map (Φ, α, γ).

Moreover the unity of the adjunction for a proof calculus $\mathcal{P} = (\mathbf{Sign}, Sen, \vdash, P, \mathrm{Pr}, \pi)$ is the map $(Id_{\mathbf{Sign}}, Id_{Sen}, \pi)$, where π is now viewed as a natural transformation $\pi\colon proofs \Rightarrow thm$.

Proof. Proposition 34 of [63]. $\square$

Lemma 4.5.2 Let $\mathcal{E} = (\mathbf{Sign}, Sen, \vdash)$ be an entailment system, $\mathcal{P}' = (\mathbf{Sign}', Sen', \vdash', P', Pr', \pi')$ be a proof calculus and $(\Phi, \alpha)\colon \mathcal{E} \to ent(\mathcal{P}')$ be a map of entailment systems. Then the following diagram is a pullback

$$\begin{tikzcd}
(\mathcal{E})^\sharp \arrow[r, "{(\Phi, \alpha, \alpha_{|thm})}"] & (ent(\mathcal{P}'))^\sharp \\
\mathcal{P} \arrow[u, "{(Id_{\mathbf{Sign}}, Id_{Sen}, \pi)}"] \arrow[r, "{(\Phi, \alpha, \gamma)}"'] & \mathcal{P}' \arrow[u, "{(Id_{\mathbf{Sign}'}, Id_{Sen'}, \pi')}"']
\end{tikzcd}$$

where $\mathcal{P} = (\mathbf{Sign}, Sen, \vdash, P, \mathrm{Pr}, \pi)$, for $P = proofs$ and $Pr = Id_{\mathbf{Set}}$, and π, γ, $proofs$ are defined by the following pullback square.

$$\begin{array}{ccc} thm & \xrightarrow{\alpha_{|thm}} & thm' \circ \Phi \\ \uparrow{\scriptstyle \pi} & & \uparrow{\scriptstyle \pi'_{\Phi}} \\ proofs & \xrightarrow[\gamma]{} & proofs' \circ \Phi \end{array}$$

Proof. Since limits in functor categories are obtained by pointwise evaluation (see e.g. [48], theorem 25.6) and **Set** has pullbacks, the pullback of the second square exists and is defined pointwise. Therefore, since every ${\pi'}_{\Phi(\Sigma,\Gamma)}$ is surjective, each $\pi_{(\Sigma,\Gamma)}$ is also surjective.[5] Thus $\mathcal{P}$ is a proof calculus.

Moreover by definition (Φ, α, γ) is a map from $\mathcal{P}$ into $\mathcal{P}'$, because $\alpha_{|thm} \circ \pi = {\pi'}_{\Phi} \circ \gamma$, and $(Id_{\mathbf{Sign}}, Id_{Sen}, \pi)$ is a map from $\mathcal{P}$ into $(\mathcal{E})^{\sharp}$, because $Id_{Sen} \circ \pi = j_{Id_{\mathbf{Sign}}} \circ \pi$. And obviously the first diagram commutes; thus it is sufficient to show that any other pair of natural transformations that makes the diagram commute factorizes in a unique way through $\mathcal{P}$.

Assume that $(\Phi, \alpha, \alpha_{|thm}) \circ (\Psi, \overline{\alpha}, \overline{\gamma}) = (Id_{\mathbf{Sign}'}, Id_{Sen'}, \pi') \circ (\Psi', \overline{\alpha}', \overline{\gamma}')$, for some $\mathcal{P}''$ and $(\Psi, \overline{\alpha}, \overline{\gamma}) \colon \mathcal{P}'' \to (\mathcal{E})^{\sharp}$, $(\Psi', \overline{\alpha}', \overline{\gamma}') \colon \mathcal{P}'' \to \mathcal{P}'$.

Since pullbacks in functor categories are computed pointwise, the following diagram is a pullback, because it is the translation of the diagram defining *proofs*, γ, π along Ψ (recalling that $\Phi \circ \Psi = \Psi'$):

$$\begin{array}{ccc} thm \circ \Psi & \xrightarrow{\alpha_{|thm_{\Psi}}} & thm' \circ \Psi' \\ \uparrow{\scriptstyle \pi_{\Psi}} & & \uparrow{\scriptstyle \pi'_{\Psi'}} \\ proofs \circ \Psi & \xrightarrow[\gamma_{\Psi}]{} & proofs' \circ \Psi' \end{array}$$

and hence, since $\alpha_{|thm_{\Psi}} \circ \overline{\gamma} = {\pi'}_{\Psi'} \circ \overline{\gamma}'$ there exists a unique $\tilde{\gamma} \colon proofs'' \Rightarrow proofs \circ \Psi$ s.t. both $\pi_{\Psi} \circ \tilde{\gamma} = \overline{\gamma}$ and $\gamma_{\Psi} \circ \tilde{\gamma} = \overline{\gamma}'$. Thus it is sufficient to show that $(\Psi, \overline{\alpha}, \tilde{\gamma})$ is a map, i.e. that $\pi_{\Psi} \circ \tilde{\gamma} = \overline{\alpha} \circ \pi''$.

Since $(\Psi, \overline{\alpha}, \overline{\gamma})$ is a map, $j_{\Psi} \circ \overline{\gamma} = \overline{\alpha} \circ \pi''$, i.e., j being the identity on *thm*, $\overline{\gamma} = \overline{\alpha} \circ \pi''$; moreover, by definition of $\tilde{\gamma}$, $\overline{\gamma} = \pi_{\Psi} \circ \tilde{\gamma}$, so that $\pi_{\Psi} \circ \tilde{\gamma} = \overline{\alpha} \circ \pi''$. □

[5] The property that the pullback of a surjective map in **Set** is also surjective is easy to check directly, and follows also (by the axiom of choice) from the general fact that in any category pullbacks of retracts are retracts, see e.g. [48], prop. 21.13

Prop. 4.5.3 The category <u>*Ent*</u> admits generalization under the functors $(\_)^+$ and *ent*.

Proof. By Theorem 4.2.1, that applies because of Prop. 4.5.1 and Lemma 4.5.2. $\square$

Remark. Note that, since maps of proof calculi do not take in account the internal structure of the proofs, any two proof calculi having the same underlying entailment system and the same functor *proofs* and natural transformation π are isomorphic. Therefore the factorization of *proofs* by $P = proofs$ and $Pr = Id_{\mathbf{Set}}$ is arbitrary. In particular this factorization forgets the proof structure and hence is not always the best possible choice, but it is the only canonical choice that can be made in the general case. $\square$

Example 4.5.4 Consider again the example of the translation of many-sorted into one-sorted logic from Example 4.4.5.

First the one-sorted entailment system $ent(\mathcal{L}_{\mathcal{L}})$ is endowed with a proof structure to get a proof calculus $\mathcal{PC}_{\mathcal{L}}$ and then, using the map of entailment system $(\Phi, \alpha): ent(\mathcal{L}_{\mathcal{L}}) \to ent(\mathcal{L}_{\mathcal{MS}})$, sketched in Example 4.4.5, a many sorted proof calculus $\mathcal{PC}_{\mathcal{MS}}$ is built, by applying the above theorem.

Since the definition of the Birkhoff's entailment system is done by induction, there is at hand a natural structure for proofs in the one-sorted framework. Consider, indeed, as proof structures just the proofs trees. Thus $\mathbf{Struct}_{\mathcal{L}}$ is the category of algebras on a tree signature (i.e. an algebraic signature including the sort *trees* and operations to define and deal with trees) and $P_{\mathcal{L}}: Th_{0\mathcal{L}} \to \mathbf{Struct}_{\mathcal{L}}$ associate with every theory (Σ, Γ) the free algebra of trees whose nodes are labeled on $Sen_{\mathcal{L}}(\Sigma)$ and s.t. each leaf is in Γ or is an equality axiom, i.e. is one among $t = t$, $t = t' \supset t' = t$, $t = t' \wedge t' = t'' \supset t = t''$, and $t_1 = t'_1 \wedge \ldots \wedge t_n = t'_n \supset f(t_1, \ldots, t_n) = f(t'_1, \ldots, t'_n)$, and for each node there exists an inference rule among weakening, instantiation and modus ponens s.t. the (label of the) node is the consequence and (the labels of) its sons are the premises of this rule. It is easy to check that, by the freeness of $P_{\mathcal{L}}(\Sigma, \Gamma)$, every theories homomorphism in $Th_{0\mathcal{L}}$ induces a homomorphic translation of proof trees, so that $P_{\mathcal{L}}$ is a functor.

Moreover $Pr_{\mathcal{L}}: \mathbf{Struct}_{\mathcal{L}} \to \mathbf{Set}$ associates every algebra with its *tree* carrier and $\pi_{\mathcal{L}}: proofs_{\mathcal{L}} \Rightarrow Sen_{\mathcal{L}}$ sends every proof tree into its root. By definition of $\vdash_B$, $\mathcal{PC}_{\mathcal{L}} = (\mathbf{Sign}_{\mathcal{L}}, Sen_{\mathcal{L}}, \vdash_B, P_{\mathcal{L}}, Pr_{\mathcal{L}}, \pi_{\mathcal{L}})$ is a proof calculus.

Thus by the above theorem, $\mathcal{PC}_{\mathcal{MS}} = (\mathbf{Sign}_{\mathcal{MS}}, Sen_{\mathcal{MS}}, \vdash, P_{\mathcal{MS}}, Pr_{\mathcal{MS}}, \pi_{\mathcal{MS}})$, where $P_{\mathcal{MS}}$ sends every many-sorted theory T to the set of pairs consisting of a theorem ϕ of T and a proof tree t for $\Phi(T)$ s.t. the root of t is $\alpha(\phi)$, $Pr_{\mathcal{MS}}$ is the identity and $\pi_{\mathcal{MS}}$ is the first projection.

Note that the proof trees in $P_{\mathcal{MS}}(T)$ may be (and in general are) labeled on one sorted sentences which are not the image of any many-sorted sentences. $\square$

4.6 Building Logical Systems

Putting together the constructions for proof calculi and for logics, a logical system can be built on top of any institution, endowing the models of a logic with an entire proof calculus inherited from another logic.

Prop. 4.6.1 Denote by $\underline{LogSys}$ the category of logical systems with maps of logical systems as arrows and by $\underline{Inst}$ the category of institutions with maps of institutions as arrows. Then the functor $(\_)^\flat \colon \underline{Inst} \to \underline{LogSys}$, defined by

- $(\mathcal{I})^\flat = (\mathbf{Sign}, Sen, Mod, \vdash_\models, thm, Id_{\mathbf{Set}}, j)$, where $\Gamma \vdash_\models \phi$ iff $A \models \phi$ for every model A s.t. $A \models \gamma \quad \forall \gamma \in \Gamma$, thm is the functor sending each theory to its theorems (validity closure), described in Section 2.2 of [63], and j is the natural subfunctor inclusion $thm \subseteq Sen$ for every institution $\mathcal{I} = (\mathbf{Sign}, Sen, Mod, \models)$ and

- $(\Phi, \alpha, \beta)^\flat = (\Phi, \alpha, \beta, \alpha_{|thm})$, where $\alpha_{|thm}$ is the restriction of α to the theorem of the domain, for every map (Φ, α, β),

is the right adjoint of the forgetful functor $inst \colon \underline{LogSys} \to \underline{Inst}$, defined by $inst(\mathcal{S}) = (\mathbf{Sign}, Sen, Mod, \models)$ for every logical system $\mathcal{S} = (\mathbf{Sign}, Sen, Mod, \vdash, \models, P, \mathrm{Pr}, \pi)$ and $inst(\Phi, \alpha, \beta, \gamma) = (\Phi, \alpha, \beta)$ for every map $(\Phi, \alpha, \beta, \gamma)$.

Moreover the unity of the adjunction for a logical system $\mathcal{S} = (\mathbf{Sign}, Sen, Mod, \vdash, \models, P, \mathrm{Pr}, \pi)$ is the map $(Id_{\mathbf{Sign}}, Id_{Sen}, Id_{Mod}, \pi)$, where π is now viewed as a natural transformation $\pi \colon proofs \Rightarrow thm$.

Proof. Propositions 37 and 31 of [63]. $\square$

Lemma 4.6.2 Let $\mathcal{I} = (\mathbf{Sign}, Sen, Mod, \models)$ be an institution, $\mathcal{S}' = (\mathbf{Sign}', Sen', Mod', \vdash', \models', P', Pr', \pi')$ be a logical system and $(\Phi, \alpha, \beta) \colon \mathcal{I} \to inst(\mathcal{S}')$ be a map of institutions. Then the following diagram is a pullback

$$\begin{tikzcd}
(\mathcal{I})^\flat \arrow[r, "{(\Phi,\alpha,\beta,\alpha_{|thm})}"] & (ent(\mathcal{S}'))^\flat \\
\mathcal{S} \arrow[u, "{(Id_{\mathbf{Sign}}, Id_{Sen}, Id_{Mod}, \pi)}"] \arrow[r, "{(\Phi,\alpha,\beta,\gamma)}"'] & \mathcal{S}' \arrow[u, "{(Id_{\mathbf{Sign}'}, Id_{Sen'}, Id_{Mod'}, \pi')}"']
\end{tikzcd}$$

where $\mathcal{S} = (\mathbf{Sign}, Sen, Mod, \vdash, \models, P, \mathrm{Pr}, \pi)$, for $\vdash$ defined by $\Gamma \vdash_\Sigma \phi$ iff both $\alpha_\Sigma(\Gamma) \vdash'_{\Phi(\Sigma,\emptyset)} \alpha_\Sigma(\phi)$ and $\Gamma \models_\Sigma \phi$, $P = proofs$ and $Pr = Id_{\mathbf{Set}}$, and π, γ, *proofs* are defined by the following pullback square.

$$\begin{tikzcd}
thm \arrow[r, "\alpha_{|thm}"] & thm' \circ \Phi \\
proofs \arrow[u, "\pi"] \arrow[r, "\gamma"'] & proofs' \circ \Phi \arrow[u, "{\pi'}_\Phi"']
\end{tikzcd}$$

Proof. Because of Lemmas 4.5.2 and 4.4.2. □

Theorem 4.6.3 The category <u>Inst</u> admits generalization under the functors $(\_)^\flat$ and *inst*.

Proof. By Theorem 4.2.1, that applies because of Prop. 4.6.1 and Lemma 4.6.2. □

Chapter 5

Structured Specification

Algebraic specifications are a tool to support the development of programs from the informal definition of the main points of the program input/output behaviour to a concrete description of a possible realization of this behaviour, through different levels of abstraction.

At each level two mechanisms are largely recognized to be essential: the capability of *structuring* specifications, because plain collections of data type and function names together with large sets of axioms on them becomes quickly unreadable and unmanageable as well as unstructured programs are, and the notion of *implementation* of a specification by another, that supports the reuse of specification modules. In this context the word "implementation" suggests the use of general functions to tailor already existing specifications to a new aim (like renaming or introduction of symbols, discarding of superfluous semantic elements or quotient of carriers, as in the paradigmatic cases of the implementation of stacks by arrays with pointer or of sets by lists) more than fixing decisional details and specializing the specification. In the practice the general operations used to relate specifications with their implementations are part of the language used to structure specifications, so that the two aspects are strictly related. Whereas the notion of implementation between two levels of abstraction mostly corresponds to the intuitive idea of tuning the behaviour of the program to be specified, for example by introducing error messages, or modifying the structure of a sub-module, for instance by requiring the submodule itself is obtained as a structured specification; in this sense implementation becomes *refinement* and, disregarding the language differences due to the different levels, can be simply represented as a model inclusion; indeed fixing more details decreases the number of satisfactory models.

Since the ultimate goal of any algebraic specification is to produce an abstract description of the programs that correctly solve a problem (hopefully in such a way that it is trivial to derive from the description one of those programs), the semantics of a specification is a class of models, but in order to be able to reason and work on specifications, a (possibly finite) syntactic description should be adopted and the reasonable candidate is the language of the expressions on the (names of the) structuring operations. In many structuring languages the operations on (the model classes of) theories yield (a specification that is denoted also by) a theory, too; this leads to consider theories as semantics of the language (with the flavor of a term algebra) and has originated a debate about what a specification is. Here it seems preferable to use the word “specification” to denote the semantics, following for example [95, 85], and “language expression” to denote the syntactic counterpart.

Due to the proliferation of algebraic formalisms, the structuring languages needs to be generalized and potentiated to deal with specifications expressed in several frameworks. A first step in this direction was made by Sannella and Tarlecki (see e.g. [84]) with the definition of *institution independent* languages to structure specifications, this concept is crucial in proving properties of composed specifications based on the structure of the specifications and disregarding the actual formalism chosen to realize the arguments. But with the goal of reusing the specifications, or also better having a library of specifications on the shelf ready to be used, a stronger notion has to be introduced, that allows to put together specifications defined in several institutions by means of the structuring operators. Simulations provide a tool to translate specifications and hence make in some sense immaterial the original framework, but to make precise the irrelevance of these translations, the structuring language (or more precisely its interpretation) has to behave uniformly on the different institutions so that the simulations are homomorphisms of this algebraic structure imposed on institutions.

In practice the different levels of abstraction a program specification goes through are, or could be better, represented by different algebraic formalisms and it seems quite a technical restriction to have a notion of implementation restrained to one institution. Simulations allow to generalize it to relate specifications defined in different institutions.

5.1 Simulations and modularity

The modularity principle requires the ability of “putting together” specifications, by means of structuring operations; moreover, following the loose approach to support stepwise refinement, any specification is a collection of its possible realizations. Thus the simulations, which deal only with the basic objects of a frame (i.e. signatures, sentences and models) have to be generalized to work on specifications, i.e. classes of models, and the compatibility with structuring operations has to be investigated.

5.1.1 Basic Specifications

Informally a specification is the collection of the admissible models of a data type; formally it is completely determined by a class of algebras over a signature.

Def. 5.1.1 Let $\mathcal{I} = (\mathbf{Sign}, Sen, Mod, \models)$ be an institution. The *specification functor*, $Spec_{\mathcal{I}}: \mathbf{Sign} \to \mathbf{Cat}^{Op}$ is the composition of Mod with the power functor, i.e.

- $Spec_{\mathcal{I}}(\Sigma)$ is the partially ordered category w.r.t. the class inclusion having as objects $\wp(|Mod(\Sigma)|)$, for all $\Sigma \in |\mathbf{Sign}|$;
- $Spec_{\mathcal{I}}(\sigma)(sp) = \{Mod(\sigma)(A) \mid A \in sp\}$ for all $\sigma \in \mathbf{Sign}(\Sigma_1, \Sigma_2)$ and all $sp \in Spec_{\mathcal{I}}(\Sigma_2)$. □

It is worth noting that the above construction is a particular case of building a new kind of objects starting from the ones explicitly given in the definition of institution, like models, signatures and so on. In the next subsection the problem will be faced from a more general point of view.

Having built a new kind of objects, a new component of the simulation dealing with them has to be defined and is called from now on μ_{Spec}, possibly decorated. As the construction of specifications relies on algebras, the modularity principle requires that μ_{Spec} is analogously based on μ_{Mod}.

Def. 5.1.2 Let $\mu: \mathcal{I} \to \mathcal{I}'$ be a simulation. The *specification simulation* associated with μ, is the partially natural transformation $\mu_{Spec}: Spec_{\mathcal{I}'} \circ \mu \to Spec_{\mathcal{I}}$ defined by: if $sp' = {\mu_{Mod_\Sigma}}^{-1} \circ \mu_{Mod_\Sigma}(sp')$, then $\mu_{Spec_\Sigma}(sp') = \mu_{Mod_\Sigma}(sp')$, else $\mu_{Spec_\Sigma}(sp')$ is undefined. If no ambiguity arises μ_{Spec_Σ} will be denoted by μ_{Spec} or, simply, by μ. □

Remark.

- The condition $sp' = {\mu_{Mod_\Sigma}}^{-1} \circ \mu_{Mod_\Sigma}(sp')$ can be rephrased as
  - $sp' \subseteq dom(\mu)_\Sigma$, that guarantees $sp' \subseteq {\mu_{Mod_\Sigma}}^{-1} \circ \mu_{Mod_\Sigma}(sp')$, and

- $A' \in sp'$ and $\mu_{Mod_\Sigma}(A') = \mu_{Mod_\Sigma}(B')$ imply $B' \in sp'$, that guarantees $sp' \supseteq {\mu_{Mod_\Sigma}}^{-1} \circ \mu_{Mod_\Sigma}(sp')$.

- Since μ is surjective on models, there exists exactly one $sp' \in Spec_{\mathcal{I}'}(\mu(\Sigma))$ for each $sp \in Spec_{\mathcal{I}}(\Sigma)$ s.t. $\mu_{Spec_\Sigma}(sp') = sp$. Such an sp' consists of $\{A' \mid \mu(A') \in sp\}$. This formalizes the intuitive idea that an institution is simulated by another if each of its specifications corresponds to one and only one specification of the other institution. It is also worth noting that the partial-naturality of μ_{Spec} guarantees the naturality of its inverse (i.e. the family of the inverse of any μ_{Spec_Σ}).

- The condition $sp' = {\mu_{Mod_\Sigma}}^{-1} \circ \mu_{Mod_\Sigma}(sp')$ required for the definedness of $\mu_{Spec_\Sigma}(sp')$ is sufficient to preserve the validity relation extended to specifications. Indeed, defining $sp \models \phi$ iff $A \models \phi$ for all $A \in sp$, if $sp' \models' \mu(\phi)$, then $\mu(sp') \models \phi$, because $A' \models' \mu(\phi)$ implies $\mu(A') \models \phi$ for any A' s.t. $\mu(A')$ is defined, by definition of simulation. But in general $\mu_{Mod_\Sigma}(sp') \models \phi$ does not imply $sp' \models' \mu(\phi)$, because A' may exist s.t. $A' \not\models '\mu(\phi)$ and $\mu(A')$ is not defined, so that the condition of validity preservation of simulations does not apply. Of course the totality of μ_{Mod_Σ} over sp', guaranteed by $sp' = {\mu_{Mod_\Sigma}}^{-1} \circ \mu_{Mod_\Sigma}(sp')$, is not necessary in the general case, but in many significant cases is needed.

The composition of the specification components associated with two composable simulations is the specification component associated with their composition.

Prop. 5.1.3 Let $\mathcal{I} = (\mathbf{Sign}, Sen, Mod, \models)$, $\mathcal{I}' = (\mathbf{Sign}', Sen', Mod', \models')$ and $\mathcal{I}'' = (\mathbf{Sign}'', Sen'', Mod'', \models'')$ be institutions and $\mu: \mathcal{I} \to \mathcal{I}'$ and $\nu: \mathcal{I}' \to \mathcal{I}''$ be simulations; then $(\nu \circ \mu)_{Spec_\Sigma} = \mu_{Spec_\Sigma} \circ \nu_{Spec_{\mu_{Sign}(\Sigma)}}$ for every signature $\Sigma \in |\mathbf{Sign}|$.

Proof. It is trivial to see that if the domains of $(\nu \circ \mu)_{Spec_\Sigma}$ and of $\mu_{Spec_\Sigma} \circ \nu_{Spec_{\mu_{Sign}(\Sigma)}}$ coincide, then $(\nu \circ \mu)_{Spec_\Sigma} = \mu_{Spec_\Sigma} \circ \nu_{Spec_{\mu_{Sign}(\Sigma)}}$. Thus it is sufficient to show that the two functions are defined on the same specifications.

Let sp'' be an $\mathcal{I}''$-specification s.t. $(\nu \circ \mu)_{Spec_\Sigma}(sp'')$ is defined; then by definition, ${\nu \circ \mu_{Mod_\Sigma}}^{-1}(\nu \circ \mu_{Mod_\Sigma}(sp'')) = sp''$, i.e. for every $A'' \in sp''$ both $\mu_{Mod_\Sigma}(\nu_{Mod_{\mu_{Sign}(\Sigma)}}(A''))$ is defined and if $\mu_{Mod_\Sigma}(\nu_{Mod_{\mu_{Sign}(\Sigma)}}(A'')) = \mu_{Mod_\Sigma}(\nu_{Mod_{\mu_{Sign}(\Sigma)}}(B''))$, then $B'' \in sp''$. Then *a fortiori* for every $A'' \in sp''$ both $\nu_{Mod_{\mu_{Sign}(\Sigma)}}(A'')$ is defined and if $\nu_{Mod_{\mu_{Sign}(\Sigma)}}(A'') = \nu_{Mod_{\mu_{Sign}(\Sigma)}}(B'')$, then $B'' \in sp''$ and hence $\nu_{Spec_{\mu_{Sign}(\Sigma)}}$ is defined on sp''. Let sp' denote $\nu_{Spec_{\mu_{Sign}(\Sigma)}}(sp'') = \{A' \mid A' = \nu_{Mod_{\mu_{Sign}(\Sigma)}}(A''), A'' \in sp''\}$; since $\mu_{Mod_\Sigma}(\nu_{Mod_{\mu_{Sign}(\Sigma)}}(A''))$ is defined for every $A'' \in sp''$, $\mu_{Mod_\Sigma}(A')$ is defined for every $A' \in sp'$ and if

$\mu_{Mod_\Sigma}(A') = \mu_{Mod_\Sigma}(B')$ for $A' = (\nu_{Mod_{\mu_{Sign}(\Sigma)}}(A'')) \in sp'$ and some B', then, since $\nu_{Mod_{\mu_{Sign}(\Sigma)}}$ is surjective, B'' exists s.t. $B' = \nu_{Mod_{\mu_{Sign}(\Sigma)}}(B'')$, so that $\mu_{Mod_\Sigma}(\nu_{Mod_{\mu_{Sign}(\Sigma)}}(A'')) = \mu_{Mod_\Sigma}(\nu_{Mod_{\mu_{Sign}(\Sigma)}}(B''))$ and hence $B'' \in sp''$, that guarantees $B' = \nu_{Mod_{\mu_{Sign}(\Sigma)}}(B'') \in sp'$. Therefore μ_{Spec_Σ} is defined on sp'.

Vice versa let $\nu_{Spec_{\mu_{Sign}(\Sigma)}}$ be defined on sp'' and μ_{Spec_Σ} be defined on $sp' = \nu_{Spec_{\mu_{Sign}(\Sigma)}}(sp'')$; then for every $A'' \in sp''$, $\nu_{Mod_{\mu_{Sign}(\Sigma)}}(A'')$ is defined and belongs to sp' so that $\mu_{Mod_\Sigma}(\nu_{Mod_{\mu_{Sign}(\Sigma)}}(A''))$ is defined too. Moreover if $\mu_{Mod_\Sigma}(\nu_{Mod_{\mu_{Sign}(\Sigma)}}(A'')) = \mu_{Mod_\Sigma}(\nu_{Mod_{\mu_{Sign}(\Sigma)}}(B''))$ for $A'' \in sp''$, then $\nu_{Mod_{\mu_{Sign}(\Sigma)}}(B'') \in sp'$, because μ_{Spec_Σ} is defined on sp', and hence $B'' \in sp''$, because $sp' = \nu_{Spec_{\mu_{Sign}(\Sigma)}}(sp'')$. $\square$

5.1.2 Structured Specifications

In order to define operations on specifications in a way consistent with a frame that allows the development of the different modules (inputs of the operations) in different institutions, an *institution independent* metalanguage is needed (see e.g. [84]).

The ST-institution independent metalanguage

The following metalanguage was introduced by Sannella and Tarlecki in [84] and is a core of basic specification building functions, carefully chosen to be able to express the more common operations of specification metalanguages, more than a friendly syntax designed to be used by profanes.

The provided operations are the following:

$<\Sigma, Ax>$ takes in input a signature Σ and a set of Σ-sentences Ax and yields the class of Σ-models that satisfy the sentences in Ax. This is the basic kind of specification in any specification building language and corresponds to the capability of axiomatize at least the elementary data types.

$\cup_{i\in I} sp_i$ takes in input an I-indexed family of specifications sp_i on the same signature Σ and yields the class of Σ-models that belong to each sp_i, i.e. its model class is the intersection of the model classes of the sp_i. By this operation the specification of a collection of symbols can be developed independently and then the wanted models are obtained as the intersection of the classes of structures where one symbol is correctly specified and the others are freely realized.

derive from sp **by** σ takes in input a specification sp on a signature Σ and a signature morphism $\sigma\colon \Sigma' \to \Sigma$ and yields the class of Σ'-models that are the translation along σ of some model of *sp*; working on standard signatures with sorts and operation symbols, this operation allows to hide and/or rename some sorts or operation symbols.

translate *sp* **by** σ takes in input a specification *sp* on a signature Σ and a signature morphism $\sigma\colon \Sigma \to \Sigma'$ and yields the class of Σ'-models whose translations along σ belong to *sp*. This operation corresponds to freely expanding the models of a specifications by any possible interpretation of the extra symbols in $\Sigma' - \Sigma$ and is very powerful in combination with the sum operation, because it allows to develop independently submodules, translate them on the union of their signature and then, using the sum, keeping only the models that restricted on any basic signature are good models of that feature.

iso close *sp* takes in input a specification *sp* on a signature Σ and yields the class of Σ-models that are isomorphic to any of the models of *sp*. Note that not only for a generic institution the validity is not required to be isomorphism independent and hence basic specifications could be non closed under isomorphism, but also the derive operation may cause the lack of closure under isomorphism.

minimal *sp* **w.r.t.** σ takes in input a specification *sp* on a signature Σ and a signature morphism $\sigma\colon \Sigma' \to \Sigma$ and yields the class of Σ-models which are minimal extensions of their σ-reducts, i.e. of all M s.t. for any submodel N of M (i.e. any domain N of a monomorphism with codomain M) if $Mod(\sigma)(N)$ and $Mod(\sigma)(M)$ are isomorphic, then M and N are isomorphic too. In the many sorted institution, for the particular case of Σ' the empty signature, the above condition corresponds to the restriction of models to the ones that do not have proper subalgebras, i.e. to the *term-generated* models. Unfortunately in both the partial and the non-strict framework this operation does not capture the common notion of term-generated, because the notion of subalgebra in those frames correspond to the categorical definition of *regular* subobject instead of (plain) subobject.

abstract *sp* **w.r.t.** Φ **via** σ takes in input a specification *sp* on a signature Σ, a signature morphism $\sigma\colon \Sigma \to \Sigma'$ and a set Φ of Σ'-sentences and yields the class of Σ-models which are Φ-equivalent to any model of *sp*, where A and B are Φ-equivalent, denoted by $A \equiv^{\Phi} B$, iff for every $\phi \in \Phi$, every A' s.t. $Mod(\sigma)(A') = A$ and every B' s.t. $Mod(\sigma)(B') = B$:

$$A' \models_{\Sigma'} \phi \iff B' \models_{\Sigma'} \phi.$$

The intuition behind the abstract operation is to enlarge the model class, by regarding as models all the structures that behave as models w.r.t. a set of sentences. The sentences are expressed on a larger signature, so that the extra-symbols can be used as variables; in the particular case of standard signature for Σ' the enrichment of Σ by some constant symbols, $A' \models_{\Sigma'} \phi$ for every A' s.t. $Mod(\sigma)(A') = A$ corresponds to the classical notion of satisfaction $A \models_{\Sigma',V} \phi$ for every valuation $V: \Sigma' - \Sigma \to A$, but in non-standard cases this operation allows to express strange and powerful requirements.

Abstracting from the example of the institution independent language of Sannella and Tarlecki, it is not difficult to intuit what is in general an institution independent metalanguage. The sorts of this metalanguage may be both *basic*, i.e. implicitly defined by the concept of institution, like the sort of signatures, of models and so on, and *derived*, i.e. built from the basic ones using categorical and set-theoretic concepts, like the specifications coming from the algebras applying the power functor. Analogously the operations of this metalanguage are defined only involving the usual categorical and set-theoretic language, so that the interpretation of sorts and operations in any institution is standard. While it is clear what a metalanguage based on categorical and set-theoretical language is, the only way to define it completely formally seems to be explicitly enumerating which sorts and operations are allowed, that clearly is unnecessary restrictive; thus the treatment here is limited to a semi-formal level and a scheme of construction of a generic metalanguage is proposed, using as paradigmatic examples the operations of [84].

A building scheme for an institution independent language

Let X be a set of variables[1], which will be evaluated in $|\mathbf{Sign}|$ for all institutions $\mathcal{I} = (\mathbf{Sign}, Sen, Mod, \models)$.

- Starting from the elements of X a set MS of *metasorts* is built, only using categorical and set-theoretical concepts; for example for any $\Sigma_1, \Sigma_2 \in X$ consider the metasort $Sign(\Sigma_1, \Sigma_2)$ of signature morphisms from Σ_1 into Σ_1.

- A set MF of metaoperations of arity in MS is built, only using categorical and set-theoretical concepts; for example for any $\Sigma \in X$ consider the metaoperation $mod: Sen(\Sigma) \to Spec(\Sigma)$, associating with any set of sentences the class of its models.

[1]Since in any known significant example, the language is based only on metavariables of sort *signatures*, here the variables are just a plain set for sake of simplicity; but there are no problems using a family of variables sets indexed on the basic elements of institutions.

- Given an institution $\mathcal{I} = (\mathbf{Sign}, Sen, Mod, \models)$ and a valuation $V: X \to |\mathbf{Sign}|$, with each symbol of MS and each operation symbol in MF the corresponding standard interpretation is associated; for example $(Sign(\Sigma_1, \Sigma_2))^{\mathcal{I},V} = \mathbf{Sign}(V(\Sigma_1), V(\Sigma_2))$ and $(mod)^{\mathcal{I},V}: Sen(\Sigma) \to Spec_{\mathcal{I}}(\Sigma)$ on Ax yields $\{A \mid A \models \alpha, \text{ for all } \alpha \in Ax\}$. □

For each new sort its translation along a generic simulation μ has to be defined, as in the case of specifications, where μ_{Spec} had been defined to translate them. Analogously to the definition of the metalanguage, in order to define the new components of simulations, symbols to denote the simulation components dealing with signatures, sentences and models, are introduced, that will be evaluated to the components of the actual simulation, and used to formally define the new components of a generic simulation by means of the categorical and set-theoretic metalanguage.

Simulation Independent Metalanguages

With the help of a metalanguage, institutions are provided of algebraic structure; thus conditions have to be investigated to guarantee that the simulations are behaving like homomorphisms of this new structure. The formulation of the condition of homomorphism is a bit complicated by the possible partiality and the countervariance of the components (like for models and specifications).

In the sequel both the symbol for the component of metasort s of a simulation and its evaluation on a concrete simulation will be denoted by μ_s.

Def. 5.1.4 Let $\mu: \mathcal{I} \to \mathcal{I}'$ be a simulation and $L = (MS, MF)$ be an institution independent metalanguage on variables X.

Then μ is an *L-homomorphism* iff for all valuations $V: X \to |\mathbf{Sign}|$ and all $op \in MF_{s_1 \ldots s_n, s}$ if a_i is related by μ to a'_i for $i = 1 \ldots n$, then $op^{\mathcal{I},V}(a_1, \ldots, a_n)$ is related by μ to $op^{\mathcal{I}',\mu \circ V}(a'_1, \ldots, a'_n)$, where two elements a and a' of the same metasort s are *related* by μ iff one of them is the image of the other one along μ_s.

Let L be an institution independent metalanguage and M be a class of simulations. Then L is *M-independent* iff μ is an L-homomorphism for each $\mu \in M$ and it is *simulation independent* iff μ is an L-homomorphism for every simulation μ. □

It is easy to check that L-homomorphisms are well behaving w.r.t. the composition of simulations, the union and the operational closure of languages, i.e. that if both $\mu: \mathcal{I} \to \mathcal{I}'$ and $\nu: \mathcal{I}' \to \mathcal{I}''$ are L-homomorphisms, then $\nu \circ \mu$ is an L-homomorphism, too, and if $\mu: \mathcal{I} \to \mathcal{I}'$ is both an L_1-homomorphism and an

L_2-homomorphism, then it is also an L'-homomorphism for both $L' = L_1 \cup L_2$ and L' the operational closure of L_i. In particular, due to the preservation of simulation independence by operational closure, by adding parameterization in λ-calculus style to a simulation independent language, a simulation independent language is obtained, too.

Of course the notion of simulation independent metalanguage is the main concept, because such a metalanguage guarantees a level of abstraction useful for defining in hierarchical way specifications, without losing the possibility of changing institution. Moreover most (all except *minimal*) of the *institution independent* operations presented in [84] are, or can be generalized to become, really simulation independent. However some quite usual operations are not simulation independent, but are M-independent for a wide class of simulations satisfying some extra-conditions.

The three institution independent operations *sum*, *basic* and *minimal* from [84], are paradigmatic examples of operations which respectively are, can be generalized to become and intrinsically are not simulation independent.

The *sum* operation, is institution independent; indeed

$$\cup_{i \in I} \mu(sp_i)^{\mathcal{I},V}$$

is the following class

$$sp = \{A \mid \text{ for every } i \in I \text{ there is } A'_i \in sp_i \text{ s.t.} A = \mu(A'_i)\};$$

Since only specifications closed w.r.t. the simulation are mapped and $\mu(A'_i) = A = \mu(A'_j)$ for every $i, j \in I$, $A'_i \in sp_j$ for every $i, j \in I$ and hence $A'_i \in \cup_{i \in I} sp_i$. Therefore

$$sp = \{A \mid \exists A' \in \cup_{i \in I} sp_i, s.t.\ A = \mu(A')\}$$

i.e.

$$\cup_{i \in I} \mu(sp_i)^{\mathcal{I},V} = \mu(\cup_{i \in I} sp_i{}^{\mathcal{I}',V \circ \mu}).$$

Also the translate is simulation independent, as it is easy (as well as boring) to check directly.

On the other hand there are operations which are not simulation independent only because their definition, as it stands, is not sufficiently powerful, but which can be rephrased in a more general way; consider for example the *basic* operation; then it is not simulation independent. Indeed the basic specification $<\Sigma, \emptyset>$ without axioms is the translation of the domain of the simulation, which is not, in general, the model class of $<\mu(\Sigma), \mu(\emptyset)>$.

More generally $<\Sigma, Ax>$ is translation of the intersection of $dom(\mu)_\Sigma$ and $<\mu(\Sigma), \mu(Ax)>$. To make the basic operation simulation independent there are at least two possibilities (the difference between them is quite a matter of taste):

- using an extra variable of sort $Spec(\Sigma)$, playing the role of the domain of the simulation:

  $Models(\_)\ in\_: Sen(\Sigma) \times Spec(\Sigma) \to Spec(\Sigma)$ is defined by

  $$Models(Ax)\ insp = \{A \mid A \in sp, A \models \alpha \text{ for all } \alpha \in Ax\};$$

- using an extra variable of sort $Funct_P[Mod(\Sigma) \to \mathbf{C}]$, the partial functor from $Mod(\Sigma)$ into a generic category $\mathbf{C}$, playing the role of the simulation itself:

  $Models(\_)\ in\ dom(\_): Sen(\Sigma) \times Funct_P[Mod(\Sigma) \to \mathbf{C}] \to Spec(\Sigma)$ is defined by

  $$Models(Ax)\ in\ dom(F) = \quad \{A \mid F(A)\ defined, A \models \alpha \text{ for all } \alpha \in Ax\}.$$

In both cases $< \Sigma, Ax >$ can be defined using the new operator:

$$< \Sigma, Ax >= Models(Ax)\ in\ Mod(\Sigma)$$

and

$$< \Sigma, Ax >= Models(Ax)\ in\ dom(Id_{Mod(\Sigma)}).$$

Note that both generalizations are simulation independent. Indeed let μ be a simulation from $\mathcal{I}$ into $\mathcal{I}'$; then it is easy to check that

$$Models(Ax)\ in\ \mu(sp)$$

is the class

$$\mu(\{A \mid A \in sp, A \models' \mu(\alpha) \text{ for all } \alpha \in Ax\}).$$

To show that also the second possibility is simulation independent, the component of simulation for the new sort $s = Funct_P[Mod(\Sigma) \to \mathbf{C}]$ has to be defined first. By definition $s^{\mathcal{I},V} = [Mod(V(\Sigma)) \to \mathbf{C}]_P$ and $s^{\mathcal{I}',\mu\circ V} = [Mod'(\mu \circ V(\Sigma)) \to \mathbf{C}]_P$; thus partial functors with domain $Mod(V(\Sigma))$ have to be translated into partial functors with domain $Mod'(\mu \circ V(\Sigma))$.

Since $\mu: Mod'(\mu \circ V(\Sigma)) \to Mod(V(\Sigma))$ is a partial functor, too, the composition with μ does the job and hence the new component μ_s is $\_ \circ \mu$. From the definition of μ_s, the simulation independence of the second version of basic specifications follows; indeed it is easy to check that

$$Models(Ax)\ in\ dom(F)$$

is the class

$$\mu(Models(\mu(Ax))\ in\ dom(F \circ \mu)).$$

Note that in both cases the auxiliary elements used to define the specification $< \Sigma, Ax >$ in terms of the simulation independent operations $Models(\_)\ in\_$ and $Models(\_)\ in\ dom(\_)$, i.e. $Mod(\Sigma)$ and $Id_{Mod(\Sigma)}$, are not expressions of the meta-language and indeed neither is simulation independent.

It is worth noting that the basic operation as defined in [84], is M-independent, for M the class of logical simulations, i.e. of surjective maps of institutions, changing, as it seems natural for such a class, the definition of the simulation component dealing with theories. Indeed logical maps translate any $\mathcal{I}$-theory th into a $\mathcal{I}'$-theory th' containing not only the translation of the axioms of th but also the axioms describing the domain of the simulation; using this translation is immediate to check the homomorphism condition.

The same techniques described to make the basic operation simulation independent can be applied to the abstract, derive and isoclose operations.

Finally the operations whose definition largely involves categorical properties of their arguments are not, nor can be made, simulation independent, because a generic simulation does not preserve categorical properties. Thus these operations can be at most M-independent for a suitable class M of simulations preserving the needed properties and hence have to be dropped when building a simulation independent language. Consider, for example, as a generalization of the *reachable* operation of ASL, the *minimal* operation of [84]. It takes in input a specification sp_1 on a signature Σ_1 together with a signature morphism σ from Σ into Σ_1 and yields in output the subclass of sp_1 of σ-*minimal* models, where a model A is σ-minimal iff it contains (to within isomorphism) no proper submodels from sp_1 with an isomorphic σ-reduct, i.e. iff any monomorphism $m: B \to A$, s.t. its σ-reduct $Mod(\sigma)(m): Mod(\sigma)(B) \to Mod(\sigma)(A)$ is an isomorphism in $Mod(\Sigma)$, is an isomorphism in $Mod(\Sigma_1)$, too. Then to have that *minimal* is preserved by a simulation, both monomorphisms and isomorphisms have to be preserved and reflected, which is not true in the general case. Therefore *minimal* is not (nor can be made) simulation independent.

5.2 Implementation

In the literature a concept of implementation is largely used in different contexts, which is based on the idea of *refinement*. So a specification sp_2 implements a specifications sp_1, denoted by $sp_1 \rightsquigarrow sp_2$, iff in sp_2 more details have been fixed and hence sp_2 has less models than sp_1, i.e. $sp_2 \subseteq sp_1$ (see e.g. [86, 95]); this concept may also be extended to functions on specifications and hence to parameterized specifications. For an introductive exposition of the subject see e.g. section 8.1 of [95].

Def. 5.2.1

- A specification sp_2 *implements* a specifications sp_1, denoted by $sp_1 \rightsquigarrow sp_2$, iff both sp_1 and sp_2 are on the same signature and $sp_2 \subseteq sp_1$.
- Let $f, g\colon Spec_{\mathcal{I}} \to Spec_{\mathcal{I}}$ be functions on specifications; then f *implements* g, denoted by $g \rightsquigarrow f$ iff $g(sp) \rightsquigarrow f(sp)$ for all specifications sp.
- Let $p_1 = \lambda X : \Sigma_{Par}.sp_1(\_)$ and $p_2 = \lambda X : \Sigma_{Par}.sp_2(\_)$ be terms of the same sort on some (institution independent) metalanguage; then p_2 *implements* p_1, denoted by $p_1 \rightsquigarrow p_2$, iff $sp_1[sp] \rightsquigarrow sp_2[sp]$ for all specifications sp. □

Prop. 5.2.2 Let sp, sp_1 and sp_2 be specifications and p_1, p_2 be parameterized specifications in a metalanguage whose specification-building operations are monotonic w.r.t. the set-inclusion.

1. If $sp \rightsquigarrow sp_1$ and $sp_1 \rightsquigarrow sp_2$, then $sp \rightsquigarrow sp_2$;
2. If $sp \rightsquigarrow sp_1$, $p_1 \rightsquigarrow p_2$ and sp is an actual parameter of p_1 (i.e. $p_1(sp)$ is defined), then $p_1(sp) \rightsquigarrow p_2(sp_1)$.

Proof. See fact. 8.1.1 of [95]. □

5.2.1 Simulations and the third dimension of implementation

Since μ_{Spec} is monotonic w.r.t. the set inclusion for all simulations μ, the implementation relation is translated by simulation from the old to the new frame. Moreover it is also preserved in the opposite direction, because the definedness of $\mu_{Spec}(sp')$ implies $sp' = {\mu_{Spec}}^{-1} \circ \mu_{Spec}(sp')$. Thus the restriction of the old implementation relation to the domain of μ_{Spec} coincides with the new implementation relation.

Prop. 5.2.3 Let $\mu\colon \mathcal{I} \to \mathcal{I}'$ be a simulation, sp'_1 and sp'_2 belong to $Spec_{\mathcal{I}'}(\mu(\Sigma))$ s.t. both $\mu(sp'_1)$ and $\mu(sp'_2)$ are defined; then $sp'_1 \rightsquigarrow sp'_2$ iff $\mu_{Spec}(sp'_1) \rightsquigarrow \mu_{Spec}(sp'_2)$.

Proof. If $sp'_1 \rightsquigarrow sp'_2$, then $sp'_2 \subseteq sp'_1$ and hence

$$\mu_{Spec}(sp'_2) = \mu_{Mod}(sp'_2) \subseteq \mu_{Mod}(sp'_1) = \mu_{Spec}(sp'_1)$$

i.e. $\mu_{Spec}(sp'_1) \rightsquigarrow \mu_{Spec}(sp'_2)$.

Vice versa if $\mu_{Spec}(sp'_1) \rightsquigarrow \mu_{Spec}(sp'_2)$, then ${\mu_{Mod}}^{-1}(\mu_{Spec}(sp'_2)) \subseteq {\mu_{Mod}}^{-1}(\mu_{Spec}(sp'_1))$, i.e., by the condition of definedness of μ_{Spec},

$$sp'_2 = {\mu_{Mod}}^{-1}(\mu_{Spec}(sp'_2)) \subseteq {\mu_{Mod}}^{-1}(\mu_{Spec}(sp'_1)) = sp'_1$$

and hence $sp'_1 \rightsquigarrow sp'_2$. □

Using simulations the concept of implementation is generalized, involving models in two institutions.

Def. 5.2.4 Let $\mu\colon \mathcal{I} \to \mathcal{I}'$ be a simulation, $sp \in Spec_{\mathcal{I}}(\Sigma)$ and $sp' \in Spec_{\mathcal{I}'}(\mu(\Sigma))$; then sp is *μ-implemented* by sp', denoted by $sp \overset{\mu}{\rightsquigarrow} sp'$, iff $sp' \subseteq dom(\mu)$ and $\mu_{Mod}(sp') \subseteq sp$.

Let $f\colon Spec_{\mathcal{I}}(\Sigma_1) \to Spec_{\mathcal{I}}(\Sigma_2)$ and $f'\colon Spec_{\mathcal{I}'}(\mu(\Sigma_1)) \to Spec_{\mathcal{I}'}(\mu(\Sigma_2))$ be functions; then f is *μ-implemented* by f', denoted by $f \overset{\mu}{\rightsquigarrow} f'$, iff $f(sp) \overset{\mu}{\rightsquigarrow} f'({\mu_{Mod}}^{-1}(sp))$ for all $sp \in Spec_{\mathcal{I}}(\Sigma)$.

Let $p_1 = \lambda X : \Sigma_{Par}.sp_1(\_)$ and $p_2 = \lambda X : \Sigma_{Par}.sp_2(\_)$ be terms of the same sort on some institution independent metalanguage; then p_1 is *μ-implemented* by p_2, denoted by $p_1 \overset{\mu}{\rightsquigarrow} p_2$, iff ${p_1}^{\mathcal{I},V} \overset{\mu}{\rightsquigarrow} {p_2}^{\mathcal{I}',\mu\circ V}$ for all valuations V for the free variables of p_1 and p_2 in $\mathcal{I}$. □

Note that in the particular case that $\mathcal{I} = \mathcal{I}'$ and μ is the identity, $\overset{\mu}{\rightsquigarrow}$ coincides with $\rightsquigarrow$ and hence every result for $\overset{\mu}{\rightsquigarrow}$ applies also to $\rightsquigarrow$.

The vertical and horizontal composability for the usual implementation relation can be generalized to deal with simulations as follows.

Prop. 5.2.5 Let $\mathcal{I}$, $\mathcal{I}'$ and $\mathcal{I}''$ be institutions, $\mu\colon \mathcal{I} \to \mathcal{I}'$ and $\nu\colon \mathcal{I}' \to \mathcal{I}''$ be simulations. The following conditions hold:

1. $sp \overset{\mu}{\rightsquigarrow} sp'$ and $sp' \overset{\nu}{\rightsquigarrow} sp''$ implies $sp \overset{\nu\circ\mu}{\rightsquigarrow} sp''$ for all $sp \in Spec_{\mathcal{I}}(\Sigma)$, all $sp' \in Spec_{\mathcal{I}'}(\mu(\Sigma))$ and all $sp'' \in Spec_{\mathcal{I}''}(\nu(\mu(\Sigma)))$.

2. $sp \overset{\mu}{\rightsquigarrow} sp'$ and $f \overset{\mu}{\rightsquigarrow} f'$ implies $f(sp) \overset{\mu}{\rightsquigarrow} f'(sp')$ for all $sp \in Spec_{\mathcal{I}}(\Sigma_1)$, all $sp' \in Spec_{\mathcal{I}'}(\mu(\Sigma_1))$ all monotonic $f\colon Spec_{\mathcal{I}}(\Sigma_1) \to Spec_{\mathcal{I}}(\Sigma_2)$ and $f'\colon Spec_{\mathcal{I}'}(\mu(\Sigma_1)) \to Spec_{\mathcal{I}'}(\mu(\Sigma_2))$.

Proof.

1. Since $sp' \subseteq dom(\mu)$ and $\nu_{Mod}(sp'') \subseteq sp'$, $sp'' \subseteq dom(\nu \circ \mu)$. Moreover if $sp' \stackrel{\nu}{\leadsto} sp''$, then $\nu_{Mod_{\mu_{Sign}(\Sigma)}}(sp'') \subseteq sp'$, and if $sp \stackrel{\mu}{\leadsto} sp'$, then $\mu_{Mod_\Sigma}(sp') \subseteq sp$, so that $\mu_{Mod_\Sigma}(\nu_{Mod_{\mu_{Sign}(\Sigma)}}(sp'')) \subseteq sp$, i.e. $sp \stackrel{\nu\circ\mu}{\leadsto} sp''$.

2. By definition of $\stackrel{\mu}{\leadsto}$, it is sufficient to show that $\mu_{Mod_\Sigma}(f'(sp')) \subseteq f(sp)$. By definition of $\stackrel{\mu}{\leadsto}$, $f \stackrel{\mu}{\leadsto} f'$ implies that $f'({\mu_{Mod_\Sigma}}^{-1}(sp_1)) \stackrel{\mu}{\leadsto} f(sp_1)$, so that $\mu_{Mod_\Sigma}(f'({\mu_{Mod_\Sigma}}^{-1}(sp_1)) \subseteq f(sp_1)$ for all $sp_1 \in Spec_{\mathcal{I}}(\Sigma_1)$; thus, for $sp_1 = \mu_{Mod_\Sigma}(sp')$, $\mu_{Mod_\Sigma}(f'({\mu_{Mod_\Sigma}}^{-1}(\mu_{Mod_\Sigma}(sp'))) \subseteq f(\mu_{Mod_\Sigma}(sp'))$. Since $sp \stackrel{\mu}{\leadsto} sp'$, $sp' \subseteq dom(\mu)$ and hence $sp' \subseteq {\mu_{Mod_\Sigma}}^{-1}(\mu_{Mod_\Sigma}(sp'))$, so that $f'(sp') \subseteq f'({\mu_{Mod_\Sigma}}^{-1}(\mu_{Mod_\Sigma}(sp')))$, because f' is monotonic. Finally from $sp \stackrel{\mu}{\leadsto} sp'$, i.e. $\mu_{Mod_\Sigma}(sp') \subseteq sp$, and from the monotony of f, the thesis follows. □

Thus a more suggestive diagram than the usual one can be proposed, where every path is an implementation (possibly via simulation) arrow and three dimensions are present: horizontally and vertically moving within an institution, while along the third dimension different institutions are connected.

$$
\begin{array}{cccccccc}
 & & & sp'_2 & p'_2 & & & p'_2(sp'_2) \\
 & \mu & \mu & & & & \mu & \\
sp_2 & p_2 & & & & p_2(sp_2) & & \\
 & & & sp'_1 & p'_1 & & & p'_1(sp'_1) \\
 & \mu & \mu & & & & \mu & \\
sp_1 & p_1 & & & & p_1(sp_1) & &
\end{array}
$$

It is worth noting the difference between this approach and the one in [16]. Indeed here implementation is defined as a relation between specifications in different institutions, while in [16] an institution is proposed whose sentences represent the implementation inside a basic institution.

Chapter 6

Conclusions and Future Work

In this thesis two main streams are coexisting: on one side, the problem of the translation of tools and results from one formalism into another one is approached, together with the analysis of the nature of relationships between formalisms; on the other side some results and tools in concrete algebraic frameworks are presented, in particular some classical logical and categorical notions for partial conditional (higher-order) specifications and for non-strict don't care algebras.

This duality of levels corresponds to a firm believe that all studies should be developed at a level as abstract as possible and that, from this point of view, the theory of institutions (general logic) is a powerful instrument to investigate the common ground behind algebraic specification formalisms and to relate and compare specifications defined in different frameworks. However details have to be fixed in order to produce concrete results, like defining inference systems or characterizing necessary and sufficient conditions for an initial object to exist. Accordingly with that principle, also future work continuating this thesis can be split in "at institution level" and "at ground algebraic level".

As institution morphisms capture the idea of building richer and more complex institutions assembling simpler one by means of categorical constructions, it would be interesting to analyze whether these constructions are compatible with the notion of implementation represented by simulations. In other words, if an institution $\mathcal{I}$ is obtained by applying an operation to some basic institutions $\mathcal{I}_1 \ldots \mathcal{I}_n$, that are simulated by some others $\mathcal{I}'_1 \ldots \mathcal{I}'_n$, then does the result of the same operation on $\mathcal{I}'_1 \ldots \mathcal{I}'_n$ simulates $\mathcal{I}$? This problem is the paraphrase of the vertical and horizontal composition for specification languages, where institutions correspond to specifications and simulations to implementation. If conditions can be found in order to have a positive answer, then a first step is taken in the direction of having a modular approach to the definition of complex institutions.

Much in the same direction, an analysis of the abstract operations needed to build institutions and possibly a collection of such operations *on the shelf* together with the properties that are preserved by them, would be useful and would clarify the expressive power of the institution formalism. In [30], starting from some concrete examples in concurrency, a preliminary attempt in this direction is proposed. But another interesting case could be, for example, the construction of the *observational* institution, where satisfaction is substituted with observational validity w.r.t. a set of experiments.

Until now simulations have been used to compare and translate results from the classical algebraic specifications field; it should be investigated whether simulations are also able to deal with more exotic problems, like the preservation of properties of concurrent systems. In order to explore this possibility the first problem is the correct definition of the involved institutions in a way that the concurrent paradigms under investigation can be naturally seen as such institutions and that the interesting properties play the role of sentences. Since in concurrency theory it is not uncommon to prove that a property is satisfied, or to add a feature to a system by some kind of translation (consider just as an example [65]), the concrete example to start from are at hand.

Another non-classical argument to test the power of simulation is the problem of the implementation of concrete data types, i.e. of individual algebras. Roughly speaking the idea is to realize a data type A on a signature Σ using an (already defined) algebra A' on some signature Σ', by representing each operation of Σ by a derived (term) operation on Σ' and by an abstraction function from A' into A. The abstraction function is surjective and partial, because elements of A' may exist that are not representing any object of A, and behaves like some sort of homomorphism (w.r.t. the higher-level signature Σ) on its domain. Depending on the rigorous definition of the informal idea of "homomorphism-like", different notion of implementation are captured. In the third chapter a correspondence between this notion of implementation and simulation is informally sketched (for the variant where the abstraction function is, in partial algebra terminology, a *strong* homomorphism), but it would be worth to investigate whether other variations of the concept of abstraction can be rephrased in institution language and to enrich the treatment using the sentence components (that in the third chapter are empty) of the institutions representing the lower-level and the higher-level algebra to preserve semantic properties, in order to refine the idea of "correct representation" of a concrete data type by another one.

An easier task is the study of the class of simulations preserving some interesting categorical properties, like the existence of terminal objects, (co)limits and such, as has been done for initiality developing the notion of categorical simulation. Analogously the analysis of the category of institutions with simulations as arrows should be developed.

On a more concrete side, the relationships between the partial paradigm and the treatment of errors and exception handling should be investigated and possibly lead to the definition of an integrated algebraic framework where non-termination, incompletely defined functions, error recovery and "true" non-strictness are available at a time, keeping distinct the logically different kinds of partiality/non-strictness.

Bibliography

[1] E. Astesiano and M. Cerioli. On the existence of initial models for partial (higher-order) conditional specifications. In *Proceedings of TAPSOFT'89*, number 351 in Lecture Notes in Computer Science, pages 74–88, Berlin, 1989. Springer Verlag.

[2] E. Astesiano and M. Cerioli. Commuting between institutions via simulation. Technical Report FMG 2, Department of Mathematics, University of Genoa, 1990.

[3] E. Astesiano and M. Cerioli. Partial higher-order specifications. In A. Tarlecki, editor, *Proceedings of Mathematical Foundation of Computer Science '91*, number 520 in Lecture Notes in Computer Science, pages 74–84, Berlin, 1991. Springer Verlag.

[4] E. Astesiano and M. Cerioli. Non-strict don't care algebras and specifications. In S. Abramsky and T.S.E. Maibaum, editors, *Proceedings of TAPSOFT'91*, number 493 in Lecture Notes in Computer Science, pages 121–142, Berlin, 1992. Springer Verlag.

[5] E. Astesiano and M. Cerioli. Non-strict don't care algebras and specifications. 1992. Submitted.

[6] E. Astesiano and M. Cerioli. Partial higher-order specifications. *Fundamenta informaticae*, 16(2):101–126, 1992.

[7] E. Astesiano and M. Cerioli. Free objects and equational deduction for partial conditional specifications. *Theoretical Computer Science*, 1993. To appear.

[8] E. Astesiano and M. Cerioli. Relationships between logical frames. In *Recent Trends in Data Type Specification*, number 655 in Lecture Notes in Computer Science, pages 126–143, Berlin, 1993. Springer Verlag.

[9] E. Astesiano, A. Giovini, G. Reggio, and E. Zucca. An integrated algebraic approach to the specification of data types, processes and objects. In M. Wirsing and J.A. Bergstra, editors, *Algebraic Methods: Theory, Tool and Applications*, number 394 in Lecture Notes in Computer Science, pages 91–116, Berlin, 1989. Springer Verlag.

[10] E. Astesiano, G.F. Mascari, G. Reggio, and M. Wirsing. On the parameterized algebraic specification of concurrent systems. In H. Ehrig, C. Floyd, M. Nivat, and J. Thatcher, editors, *Proc. TAPSOFT'85, Vol. 1*, number 185 in Lecture Notes in Computer Science, pages 342–358, Berlin, 1985. Springer Verlag.

[11] E. Astesiano, C. Bendix Nielsen, N. Botta, A. Fantechi, A. Giovini, P. Inverardi, E. Karlsen, F. Mazzanti, J. Storbank Pedersen, G. Reggio, and E. Zucca. The draft formal definition of Ada. Technical report, CEC MAP project: The Draft Formal Definition of ANSI/STD 1815A Ada, 1986.

[12] E. Astesiano and G. Reggio. An outline of the SMoLCS approach. In M. Venturini Zilli, editor, *Mathematical Models for the Semantics of Parallelism, Proc. Advanced School on Mathematical Models of Parallelism*, number 280 in Lecture Notes in Computer Science, pages 81–113, Berlin, 1987. Springer Verlag.

[13] E. Astesiano and G. Reggio. SMoLCS-driven concurrent calculi. In H. Ehrig, R. Kowalski, G. Levi, and U. Montanari, editors, *Proc. TAPSOFT'87, Vol. 1*, number 249 in Lecture Notes in Computer Science, pages 169–201, Berlin, 1987. Springer Verlag.

[14] E. Astesiano and G. Reggio. A structural approach to the formal modelization and specification of concurrent systems. Technical Report 1, Dipartimento di Informatica e Scienze dell'Informazione, Università di Genova, Genova, Italy, 1992.

[15] J. Barwise. Axioms for abstract model theory. *Annals of Mathematical Logic*, 7:221–165, 1974.

[16] C. Beierle and A. Voss. Viewing implementations as an institution. In D.H. Pitt, A. Poigné, and D.E. Rydeheard, editors, *Proceedings of Category Theory and Computer Science*, number 283 in Lecture Notes in Computer Science, pages 196–218, Berlin, 1987. Springer Verlag.

[17] J.A. Bergstra and J.V. Tucker. The inescapable stack: an exercize in algebraic specification with total functions. Technical Report P8804, University of Amsterdam; Programming Research Group, 1988.

[18] G. Bernot, M. Bidoit, and C. Choppy. Abstract data types with exception handling: an initial approach based on the distinction between exceptions and errors. *Theoretical Computer Science*, 46(1):13–45, 1986.

[19] G. Birkhoff. On the structure of abstract algebras. In *Proceedings of Cambridge Philosophiae Society*, volume 31, pages 433–154, 1935.

[20] E. Börger. *Computability, Complexity, Logic.* Number 128 in Studies in Logic and the Foundations of Mathematics. North Holland, Amsterdam, 1989.

[21] M. Broy, C. Pair, and M. Wirsing. A systematic study of models of abstract data types. *Theoretical Computer Science*, 33:137–174, 1984.

[22] M. Broy and M. Wirsing. Partial abstract types. *Acta Informatica*, 18, 1982.

[23] M. Broy and M. Wirsing. On the algebraic specifications of finitary infinite communicating sequential processes. In *Proceedings of IFIP TC2 Working Conference on Formal Description of Programming Concepts II*, Amsterdam, 1983. North Holland.

[24] M. Broy and M. Wirsing. Generalized heterogeneous algebras and partial interpretations. In *Proceedings of CAAP'83*, number 159 in Lecture Notes in Computer Science, pages 1–34, Berlin, 1984. Springer Verlag.

[25] M. Broy and M. Wirsing. Ultra-loose algebraic specifications. *Bulletin EATCS*, 35:117–128, 1988.

[26] P. Burmeister. *A Model Theoretic Oriented Approach to Partial Algebras.* Akademie Verlag, Berlin, 1986.

[27] R.M. Burstall and J. A. Goguen. Putting theories together to make specifications. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, pages 1045–1058, Cambridge, 1977.

[28] R.M. Burstall and J. A. Goguen. The semantics of clear, a specification language. In D. Bjørner, editor, *Proceedings of the 1979 Copenhagen Winter School on Abstract Software Specification*, number 86 in Lecture Notes in Computer Science, pages 292–332, Berlin, 1980. Springer Verlag.

[29] M. Cerioli. A sound and equationally-complete deduction system for partial conditional (higher order) types. In *Proceedings of the 3rd Italian Conference of Theoretical Computer Science*. World Scientific, 1989.

[30] M. Cerioli and G. Reggio. Institutions for very abstract specifications. Draft, presented at the 9th ADT Workshop, 1992.

[31] G. Costa and G. Reggio. Abstract dynamic data types: a temporal logic approach. In A. Tarlecki, editor, *Proceedings of Mathematical Foundation of Computer Science '91*, number 520 in Lecture Notes in Computer Science, pages 103–112, Berlin, 1991. Springer Verlag.

[32] R. Diaconescu, J.A. Goguen, and P. Stefaneas. Logical support for modularization. Draft, 1991.

[33] H. Erigh, M. Baldamus, and F. Cornelius. Theory of algebraic module specification including behavioural semantics, constraints and aspects of generalized morphisms. In *Proceedings of 2nd International Conference on Algebraic Methodology and Software Technology*, pages 101–125, Iowa City, Iowa, USA, 1991.

[34] H. Erigh and B. Mahr. *Fundamentals of Algebraic Specifications 1: Equations and Initial semantics*, volume 6 of *EATCS Monographs on Theoretical Computer Science*. Springer Verlag, New-York, 1985.

[35] S. Feferman. A new approach to abstract data types, I informal development. *Mathematical Structures in Computer Science*, 2:193–229, 1992.

[36] J. Fiadeiro and A. Sernadas. Structuring theories on consequence. In D. Sannella and A. Tarlecki, editors, *Recent Trends in Data Type Specification*, number 332 in Lecture Notes in Computer Science, pages 44–72, Berlin, 1987. Springer Verlag.

[37] M. Gogolla. Partially ordered sorts in algebraic specifications. In *Proceedings CAAP'83*, number 159 in Lecture Notes in Computer Science, pages 139–153, Berlin, 1984. Springer Verlag.

[38] M. Gogolla. On parametric algebraic specifications with clean error handling. In *Proceedings TAPSOFT'87*, number 249 in Lecture Notes in Computer Science, pages 81–95, Berlin, 1987. Springer Verlag.

[39] J. Goguen and J. Meseguer. Order-sorted algebra I: Equational deduction for multiple inheritance, overloading, exceptions and partial operations. Technical Report SRI-CSL-89-10, Computer Science Lab., SRI International, 1989.

[40] J. Goguen and J. Meseguer. Order-sorted algebra I: Equational deduction for multiple inheritance, overloading, exceptions and partial operations. *Theoretical Computer Science*, 1992. To appear.

[41] J. Goguen, J. Thatcher, and Wagner. An initial algebra approach to the specification, correctness, and implementation of abstract data types. In R. Yeh, editor, *Current Trends in Programming Methodology*, pages 80–149. Prentice-Hall, 1976.

[42] J. Goguen, J. Thatcher, E. Wagner, and J. Wright. Abstract data types as an initial algebra and correctness of data representation. In *Proceedings of Conference on Computer Graphics, Pattern Recognition and Data Structure*, pages 89–93, 1976.

[43] J.A. Goguen. Abstract errors for abstract data types. In J. Dennis, editor, *Proceedings of Conference on Formal Description of Programming Concepts*, Amsterdam, 1977. North Holland.

[44] J.A. Goguen and R.M. Burstall. Introducing institutions. In E. Clarke and D. Kozen, editors, *Logics of Programs Workshop*, number 164 in Lecture Notes in Computer Science, pages 221–256, Berlin, 1984. Springer Verlag.

[45] J.A. Goguen and R.M. Burstall. A study in the foundations of programming methodology: Specifications, institutions, charter and parchments. In D. Pitt, S. Abramsky, A. Poigné, and D. Rydehard, editors, *Proceedings of Summer Workshop on Category Theory and Computer Programming*, number 240 in Lecture Notes in Computer Science, pages 313–333, Berlin, 1986. Springer Verlag.

[46] J.A. Goguen and R.M. Burstall. Institutions: Abstract model theory for specification and programming. *Journal of the Association for Computing Machinery*, 39(1):95–146, 1992.

[47] J.A. Goguen and R. Diaconescu. A survey of order sorted algebra. Draft, 1992.

[48] H. Herrlich and G.E. Strecker. *Category Theory, an Introduction.* Heldermann Verlag, Berlin, 1979.

[49] C.A.R. Hoare. Proof of correctness of data representation. *Acta Informatica*, 1:271–281, 1972.

[50] G. Huet and D. Oppen. Equations and rewrite rules: a survey. In *Formal Language Theory: Perspectives and Open Problems*. New York, 1980.

[51] H.J. Keisler. *Model Theory for Infinitary Logic*. North Holland, New-York, 1971.

[52] F.W. Lawvere. Adjointness in foundations. *Dialectica*, 23:281–296, 1969.

[53] F.W. Lawvere. Equality in hyperdoctrine and comprehension schema as an adjoint functor. In *Proceedings of American Mathematical Society*, pages 1–14, 1970.

[54] S. MacLane. *Categories for the Working Mathematician*. Springer Verlag, 1971.

[55] B. Mahr and J.A. Makowsky. An axiomatic approach to semantics of specification languages. In *Proceedings of the 6th GI-Conference on Theoretical Computer Science*, number 145 in Lecture Notes in Computer Science, Berlin, 1984. Springer Verlag.

[56] B. Mahr and J.A. Makowsky. Characterizing specification languages which admit initial semantics. In *Proceedings CAAP'83*, number 159 in Lecture Notes in Computer Science, pages 300–316, Berlin, 1984. Springer Verlag.

[57] B. Mahr and J.A. Makowsky. Characterizing specification languages which admit initial semantics. *Theoretical Computer Science*, 31:49–59, 1984.

[58] V. Manca and A. Salibra. Soundness and completeness of the birkhoff equational calculus for many-sorted algebras with possibly empty carriers. *Theoretical Computer Science*, 94(1):101–124, 1992.

[59] V. Manca, A. Salibra, and G. Scollo. Equational type logic. *Theoretical Computer Science*, 77:131–159, 1990. Special Issue dedicated to AMAST'89.

[60] V. Manca, A. Salibra, and G. Scollo. On the expressiveness of equational type logic. In C.M.I. Rattray and R.G. Clark, editors, *The Unified Computation Laboratory*. Oxford University Press, 1992.

[61] B. Mayoh. Galleries and institutions. Technical Report DAIMI PB - 191, AArhus University, 1985.

[62] K. Meinke. Universal algebra in higher types. *Theoretical Computer Science*, 100(2):385–417, 1992.

[63] J. Meseguer. General logics. In *Logic Colloquium '87*, pages 275–329, Amsterdam, 1989. North Holland.

[64] J. Meseguer and J. Goguen. Initiality, induction and computability. In M. Nivat and J. Reynolds, editors, *Algebraic Methods in Semantics*, pages 459–540. Cambridge, 1985.

[65] R. Milner. Interpreting one concurrent calculus in another. *Theoretical Computer Science*, 75:3–13, 1990.

[66] B. Möller. Algebraic specification with higher-order operations. In *Proceedings of IFIP TC 2 Working Conference on Program Specification and Transformation*, Amsterdam, 1987. North Holland.

[67] B. Möller, A. Tarlecki, and M. Wirsing. Algebraic specification with built-in domain constructions. In M. Dauchet and M. Nivat, editors, *Proceedings of CAAP'88*, number 299 in Lecture Notes in Computer Science, pages 132–148, Berlin, 1988. Springer Verlag.

[68] P. Mosses. Unified algebras and institutions. In *Proceedings of 4th Annual IEEE Symposium on Logic in Computer Science*, pages 304–312, 1989.

[69] P.D Mosses. Unified algebras and modules. Technical Report DAIMI PB-266, CS Dept., Aarhus University, 1988.

[70] M. Navarro, P. Nivela, F. Orejas, and A. Sanchez. On translating partial to total specifications with applications to theorem proving for partial specifications. Technical Report LSI-89-21, Universitat Politecnica de Catalunya, 1990.

[71] P. Padawitz and M. Wirsing. Completeness of many-sorted equational logic revisited. *Bulletin EATCS*, 24, 1984.

[72] D.L. Parnas. A technique for software module specification. *Communications of A.C.M.*, 15, 1972.

[73] A. Poigné. Another look at parametrization using algebraic specifications with subsort. In M.P. Chytil and V. Koubek, editors, *Proceeding of Mathematical Foundations of Computer Science'84*, number 176 in Lecture Notes in Computer Science, pages 471–479, Berlin, 1984. Springer Verlag.

[74] A. Poigné. Partial algebras, subsorting, and dependent types: Prerequisites of error handling in algebraic specifications. In *Recent Trends in Data Type Specification*, number 332 in Lecture Notes in Computer Science, pages 208–234, Berlin, 1987. Springer Verlag.

[75] A. Poigné. Foundations are rich institutions, but institutions are poor foundations. 1988.

[76] A. Poigné. Foundations are rich institutions, but institutions are poor foundations. In H. Ehrig, H. Herrlich, Kreowski H. J., and G. Preuß, editors, *Categorical Methods in Computer Science*, number 393 in Lecture Notes in Computer Science, pages 82–101, Berlin, 1989. Springer Verlag.

[77] Z. Qian. Higher-order order-sorted algebras. In G. Goos and J. Hartmanis, editors, *Proceedings 2nd International Conference on Algebraic and Logic Programming*, number 463 in Lecture Notes in Computer Science, pages 86–100, Berlin, 1990. Springer Verlag.

[78] G. Reggio. Entities: an istitution for dynamic systems. In H. Ehrig, K.P. Jantke, F. Orejas, and H. Reichel, editors, *Recent Trends in Data Type Specification*, number 534 in Lecture Notes in Computer Science, pages 246–265, Berlin, 1991. Springer Verlag.

[79] G. Reggio. Event logic for specifying abstract dynamic data types. In *Recent Trends in Data Type Specification*, number 655 in Lecture Notes in Computer Science, Berlin, 1992. Springer Verlag.

[80] H. Reichel. *Initial Computability, Algebraic Specifications, and Partial Algebras.* Akademie Verlag, 1986.

[81] A. Salibra and G. Scollo. A soft stairway to institutions. In *Recent Trends in Data Type Specification*, number 655 in Lecture Notes in Computer Science, pages 310–329, Berlin, 1992. Springer Verlag.

[82] D. Sannella, S. Sokolowski, and A. Tarlecki. Toward formal development of programs from algebraic specifications: Parameterization revisited. *Acta Informatica*, 1992. To appear.

[83] D. Sannella and A. Tarlecki. On observational equivalence and algebraic specifications. *Journal of Comp. and Sys. Sciences*, 34:150–178, 1987.

[84] D. Sannella and A. Tarlecki. Specifications in an arbitrary institution. *Information and Computation*, 76:165–210, 1988.

[85] D. Sannella and A. Tarlecki. Toward formal development of programs from algebraic specifications: Model-theoretic foundations. In W. Kuich, editor, *Proceedings of 19th International Colloquium on Automata, Languages and Programming*, number 623 in Lecture Notes in Computer Science, pages 656–671, Berlin, 1992. Springer Verlag.

[86] D. Sannella and M. Wirsing. A kernel language for algebraic specification and implementation. In M. Karpinski, editor, *International Conference on Foundations of Computation*, number 158 in Lecture Notes in Computer Science, pages 413–427, Berlin, 1983. Springer Verlag.

[87] R.A.G. Seely. Hyperdoctrines, natural deduction, and the Beck condition. *Zeitschrift für Math. Logic und Grundlagen der Math.*, 1984.

[88] G. Smolka. Order-sorted horn logic: Semantics and deduction. Technical Report SEKI SR-86-17, Fachbereich Informatik, Univerisität Kaiserslautern, 1986.

[89] A. Tarlecki. Free construction in algebraic institutions. In M.P. Chytil and V. Koubek, editors, *Proceedings of Mathematical Foundation of Computer Science '84*, number 176 in Lecture Notes in Computer Science, pages 526–534, Berlin, 1984. Springer Verlag.

[90] A. Tarlecki. On the existence of free models in abstract algebraic institutions. *Theoretical Computer Science*, 37(3):269–304, 1985.

[91] A. Tarlecki. Bits and pieces of the theory of institutions. In D. Pitt, S. Abramsky, A. Poigné, and D. Rydehard, editors, *Proceedings of Summer Workshop on Category Theory and Computer Programming*, number 240 in Lecture Notes in Computer Science, pages 334–360, Berlin, 1986. Springer Verlag.

[92] A. Tarlecki. Quasi-varieties in abstract algebraic institutions. *Journal of Computer and System Science*, 33, 1986.

[93] A. Tarlecki. Institution representation. draft note, November 1987.

[94] A. Tarski. Fundamentale begriffe der methodologie der deduktiven wissenschaften. *Logique, Sémantique, Métamathématique*, 1:67–116, 1972. French translation.

[95] M. Wirsing. Algebraic specification. In *Handbook of Theoretical Computer Science*. North Holland, 1990.