GENERALIZED GENERAL LOGICS

ROBERT HELGESSON

Doctor of Philosophy (PhD)
Department of Computing Science
Umeå University

May 2013

ISBN (digital): 978-91-7459-606-9
ISBN (printed): 978-91-7459-605-2
ISSN: 0348-0542
UMINF: 13.07
Electronic version available at http://umu.diva-portal.org/
Printed by: Print & Media, Umeå University
Umeå, Sverige, 2013

ABSTRACT

Logic as a vehicle for sound reason has a long and lustrous history and while most developments follow the traditional notions of binary truth and crisp sentences, great efforts have been placed into the problem of reasoning with uncertainties. To this end the field of "fuzzy logic" is now of great importance both theoretically and practically. The present monograph seeks to extend and clarify the treatment of non-classical notions of logic and, more broadly, information representation in general. This is done using two theoretical developments presented with additional discussions concerning possible applications.

The first theoretical development takes the form of a novel and strictly categorical term monad that readily allows for a multitude of non-classical situations and extensions of the classical term concept. For example, using monad composition we may represent and perform substitutions over many-valued sets of terms and thereby represent uncertainty of information. As a complementary example, we may extend this term monad to incorporate uncertainty on the level of variables and indeed the operators themselves. These two notions are embodied in the catchphrases "computing with fuzzy" and "fuzzy computing".

The second theoretical development is a direct generalization of the notion of *general logics*, a successful categorical framework to describe and interrelate the various concepts included under the umbrella term of 'logic'. The initial leap towards general logics was the introduction of *institutions* by Goguen and Burstall. This construction cover the semantic aspects of logics and in particular, axiomatizes the crucial satisfaction relation. Adding structures for, e. g., syntactic entailment and proof calculi, Meseguer established general logics as a framework capable of describing a wide range of logics. We will in our generalization further extend general logics to more readily encompass non-classical notions of truth such as in fuzzy logics and logics operating over non-classical notions of sets of sentences.

SAMMANFATTNING

Logik är ett gammalt och kraftfullt verktyg för att föra korrekta resonemang och människor har under lång tid studerat och tillämpat olika logiska system. Dessa studier har till stor del fokuserat på logik som innefattar ett binärt sanningsbegrepp, det vill säga logik där en utsaga är antingen sann eller falsk. Givetvis är detta en väldigt begränsad vy då vi i vår vardag ofta måste värdera sanning på en betydligt mer flexibel skala. På samma sätt måste vi ofta resonera och fatta beslut utifrån information som på olika sätt kan vara osäker. En djupare förståelse för dessa *otraditionella logiker* är därför av stort intresse till exempel i expertsystem som mekaniskt föreslår beslut utifrån osäker information. I denna monografi beskriver vi två matematiska ramverk som tillåter oss att beskriva dessa typer av otraditionella logiker och otraditionell representation av information i allmänhet. Dessa teoretiska konstruktioner är kompletterade med diskussioner relaterade till praktiska och vidare teoretiska tillämpningar.

Den första av de två teoretiska konstruktionerna är en strikt kategoriteoretiskt definierad termmonad som kan beskriva termer och variabelsubstitution för en mängd olika otraditionella termkonstruktioner. Till exempel, genom monadkomposition kan vi representera flervärda mängder av termer och detta ger oss möjlighet att representera osäker information. Alternativt kan vi konstruera en termmonad som uttrycker osäkerhet på ett mer fundamental sätt genom att termernas ingående variabler och operatörer i sig kan ha associerad osäkerhet.

Den andra teoretiska konstruktionen är en direkt generalisering av det matematiska ramverket "generell logik" som axiomatiserar de olika komponenter som ofta faller under logikbegreppet. I generell logik beskrivs dock dessa utifrån en traditionell vy och för att beskriva otraditionella logiker krävs därför ofta obekväma representationer. Den nämnda generaliseringen syftar därför till att förenkla konstruktionen av mer exotiska logiker inom ett ramverk som till stor del liknar generell logik.

PUBLICATIONS

Some passages and figures have appeared previously in the following publications:

Patrik Eklund, Robert Helgesson, and Helena Lindgren. Towards refinement of clinical evidence using general logics. In L. Rutkowski, R. Tadeusiewicz, L. A. Zadeh, and J. M. Zurada, editors, *ICAISC*, volume 5097 of *Lecture Notes in Computer Science*, pages 1029–1040. Springer, 2008. ISBN 978-3-540-69572-1.

Patrik Eklund and Robert Helgesson. Composing partially ordered monads. In R. Berghammer, A. Jaoua, and B. Möller, editors, *RelMiCS*, volume 5827 of *Lecture Notes in Computer Science*, pages 88–102. Springer, 2009. ISBN 978-3-642-04638-4. doi: 10.1007/978-3-642-04639-1_7.

Patrik Eklund and Robert Helgesson. Monadic extensions of institutions. *Fuzzy Sets and Systems*, 161(18):2354–2368, 2010. ISSN 0165-0114. doi: 10.1016/j.fss.2010.03.002.

Patrik Eklund, M.Ángeles Galán, Robert Helgesson, and Jari Kortelainen. Fuzzy terms. *Fuzzy Sets and Systems*, 2013. ISSN 0165-0114. doi: 10.1016/j.fss.2013.02.012.

Patrik Eklund, M.Ángeles Galán, Robert Helgesson, and Jari Kortelainen. From Aristotle to Lotfi. In R. Seising, E. Trillas, C. Moraga, and S. Termini, editors, *On Fuzziness*, volume 298 of *Studies in Fuzziness and Soft Computing*, pages 147–152. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-35640-7. doi: 10.1007/978-3-642-35641-4_23.

Patrik Eklund, Mario Fedrizzi, and Robert Helgesson. Monadic social choice. In A. G. S. Ventre, A. Maturo, S. Hosková-Mayerová, and J. Kacprzyk, editors, *Multicriteria and Multiagent Decision Making with Applications to Economics and Social Sciences*, volume 305 of *Studies in Fuzziness and Soft Computing*. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-35634-6.

ACKNOWLEDGMENTS

I first and foremost express my gratitude to my thesis supervisor Patrik Eklund whose support, endless enthusiasm, and deep knowledge has been invaluable to me throughout the process of understanding and appreciating the power and beauty that arise from adopting strict formalisms in the field of computer science.

Further, I have through the years as a PhD student collaborated with a number of excellent scientists such as Mario Fedrizzi, M.Ángeles Galán, Jari Kortelainen, and Helena Lindgren. As collaborators on articles and conference presentations they have all had a direct and influential effect on me and this text. A special mention goes to Ulrich Höhle whose insights into monoidal categories and quantales, shared with me and Patrik during numerous online meetings, have greatly improved this thesis. More generally I would like to thank all colleagues I have met at conferences, workshops, and other functions for the many interesting and inspiring discussions we have had.

I finally also would like to express my appreciation of the department of computing science at Umeå University whose support and expertise has been crucial in me being able to complete my studies.

CONTENTS

LIST OF FIGURES

1 INTRODUCTION

The concept of mathematical logic certainly has a long and lustrous history. From its dawn primarily in ancient Greece to its development into a field of modern rigorous study in the 1800's, it has formed a foundation on which many great works of mathematics and computer science have been constructed [37]. As a result of the attention paid to the topic, there exists today not a single all-encompassing notion of logic but a wide variety of logics for all imaginable purposes. For the field of computer science, the situation has been succinctly summarized by the well known quote:

> There is a population explosion among the logical systems being used in computer science. Examples include first order logic (with and without equality), equational logic, Horn clause logic, second order logic, higher order logic, infinitary logic, dynamic logic, process logic, temporal logic, and modal logic; moreover, there is a tendency for each theorem prover to have its own idiosyncratic logical system.
>
> – Goguen and Burstall [70]

With all these possible logics, it is clearly of interest to view the nature of these logics and their relationships as a mathematical field of study in and of itself. To accomplish this it is of essential importance to have good definitions that closely match our intuition of the nature of logic. The solution brought forth by Goguen and Burstall [70] was the notion of *institutions*, a notion that we essentially may view as an axiomatization of the $\models$ symbol from logic. A logic described within this axiomatic framework will have to provide a strict construction of the models used in the logic as well as the structure of the sentences used to write formulas in the logic. Finally, the institutional definition of the logic

includes the aforementioned $\models$ symbol as a relation allowing us to determine when a given sentence holds in the logic.

As described by Diaconescu [35] in his excellent historical review of institution theory, the concept of institutions first appeared in Burstall and Goguen [26] under the name of 'language' but the modern name soon became popular and by the time of the first major treatment of institution theory in its own right [see 70] it was a well established name. Burstall and Goguen were deeply interested in the use of institutions as a foundation for specification theory and, e. g., the wide spread specifications languages CafeOBJ [34] and CACL [7] both have foundations that to a large degree have been influenced by institution theory [35]. Once the worth of a framework of this kind became recognized, further developments and applications were inevitable and fruitful work has, e. g., arisen in model theory.

We will here, however, focus on a framework different from but very much inspired by institution theory. This framework, called *general logics*, arose from the desire of a formal foundation for logic programming. Noting that institutions were biased in favor of model theory, which is insufficient to fully understand logic programming, Meseguer [105] built general logics such that it encompasses both the model theoretic and proof theoretic aspects of logic. His construction is modularized and actually includes institutions as an essential component, namely the one providing a logic's semantic notion of entailment. Beyond this it also includes, e. g., a component that axiomatize syntactic entailment, provided by the $\vdash$ relation, and a component that axiomatize the proof calculus. Consequently, general logics puts into concrete mathematical terms the elusive concepts that make up what we typically call 'logic'. In Helgesson [77] we attempted a self-contained and accessible presentation of general logics as described by Meseguer [105] and Loeckx et al. [98].

When exploring the definitions of general logics is it possible to make two broad but important observations: On the one hand one may say that general logics is, in certain aspects, an opinionated framework. For example, it is possible within general logics to describe and work with a great many logics but increasing levels of cleverness are necessary the further you stray from the or-

thodoxy of a world of trues and falses into the world of maybes and sortofs. On the other hand, general logics remains deeply neutral concerning other matters. In particular, general logics says little to nothing about the actual structure of sentences and models. While this, in many cases, is a quite desirable property for a logic framework, we nevertheless must recognize that great swathes of logics have in common a deep dependency on terms.

In this thesis, we will address the first point by presenting a generalization of general logics – up to proof calculi – that gives us a framework suitable for descriptions of non-classical logic with great freedom as to the choice of, e. g., notions of truth and representations of sentence structures. Having such a generalized general logic we then address the second point by providing a specialization for logics over terms. This specialization we call *substitution logic* as the concepts of variables and variable substitution is shared by all such logics. It is worth noting that by 'terms' we here will consider a notion defined in a considerably more rigorous manner than traditionally employed. In particular, it is for a sufficiently general construction of substitution logic not adequate to rely on the definition of terms commonly encountered in the literature as such terms do not readily generalize to non-classical constructions of term sets or terms over non-classical signatures.

By traditional terms we will consider any construction that is in essence 'verbal', i. e., one that is equivalent to a text of the form:

> For a signature Ω and variable set X we have that all x in X are terms, all constants in Ω are terms, and for an n-ary operator ω in Ω we have $\omega(t_1, \ldots, t_n)$ a term if $t_1, \ldots, t_n$ are terms.

There are, for our purposes, certain crucial weaknesses in such a definition. For example, it does not carry variable substitutions as an inherent functionality – instead substitutions are added later, typically using a similarly verbal inductive definition. This makes it difficult to treat the term construction in a category theoretical framework, specifically, it is hard to show functoriality. As another example, imagine we want to represent uncertain infor-

mation in the form of fuzzy sets of terms, in this case we again run into trouble with respect to substitutions, in particular, how should we interpret the definition above when wishing to substitute a variable by a fuzzy set of terms? As a final example we observe that the verbal definition provide us with little support if we wish to entertain the idea of generalizing signatures such that, e. g., an operator may have an associated uncertainty.

Together these weaknesses cause the informal definition above to be rather uninformative with respect to possible generalizations that may be of interest in a general framework for substitution logics. For the underlying terms we will therefore present a strict category theoretical construction, building on previous work of Eklund et al. [48] but extended to many-sortedness, i. e., we will here focus on terms that are inherently discriminated into different *sorts* – or *types* as they are typically known as in computer science. Such discrimination is of importance in many practical applications and allow us to, e. g., talk of Boolean or integer terms and operators such as $\leqslant$ that takes integer arguments but whose result is a Boolean. The step taken when generalizing from the unsorted term construction to the many-sorted construction is often thought of as a trivial one but as will be seen, this is not the case. The misconception as to the simplicity of this generalization may have its root in the verbal term construction above, where it is indeed the case that the generalization is simple.

As already alluded to, we will in this text rely a great deal on category theory as a foundation on top of which we define the necessary constructions. This brings us several advantages for free, in particular we are given a wide existing vocabulary suitable for the task at hand. Briefly stated, category theory is a field of mathematics that allow rigorous study of mathematical structures and transformations between them in a consistent and convenient manner. We will also make extensive use of certain algebraic structures and notions from universal algebra. So as to make this text relatively self-contained, these underlying concepts will in Chapter 2 be briefly presented in a way that hopefully is clear and accessible to readers unfamiliar with category theory and the algebraic notions that will be employed. Clearly, though, it is a considerable advantage to have a prior solid un-

derstanding of category theory and algebra. Prior knowledge of the basic notions of discrete mathematics and mathematical logic is, however, assumed. Though, for some easily forgotten details, brief descriptions have been included.

The structure of this document is as follows – note that we include brief explanations of the contributions given in the various sections.

As already mentioned Chapter 2 provides a brief coverage of the necessary basic notions of category theory and algebra. In Chapter 2 we also introduce a novel notion of many-sorted signatures as well as their corresponding notion of terms. In this chapter we also introduce some results concerning monads: Firstly, we show that the notion of partially ordered monads may be extended in a natural way to *Kleene monads* that induce Kleene algebras in an interesting way. Secondly, we show a simple but rather interesting result identifying the Kleisli category of a composed monad with a "Kleisli over Kleisli" category.

Next, Chapter 3 will provide a relatively thorough treatment of classical general logic. No new constructions or results are presented in this chapter.

In Chapter 4, however, we introduce the generalized constructions over general logics so as to admit convenient treatment of non-classical logics. The notion of substitution logic is defined and discussed in that same chapter.

A number of examples of possible applications within the field of computer science and engineering are introduced in Chapter 5. Similarly Chapter 6 introduces an application within the field of health care. Finally, Chapter 7 will conclude the monograph with a brief discussion and prospects of future work.

2 CATEGORY THEORY AND ALGEBRA

From its first embryonic developments by Eilenberg and Mac Lane in the first half of the 1940's – with their seminal article *General theory of natural equivalences* published in 1945 [41], being the major introduction – category theory has developed into a field of mathematical study in its own right. Then, certain features of category theory known as functors and natural transformations were developed as a means to solve problems related to group theory and topology, in particular one was concerned about how transitions among mathematical structures could be understood. Today, category-theoretic notions have been applied in many different areas of mathematics and computer science and, as a result, a large number of elegant results have been produced. It has even been a useful tool in the study of the foundations of mathematics, more specifically it may be considered an alternative to axiomatic set theory as the basis of our understanding of mathematics. The work of Lawvere has been particularly prominent in this area [14, 16, 95, 96]. We will here not assume such foundations, though, instead we rely on traditional set theory, a la Zermelo–Fraenkel set theory with axiom of choice (ZFC).

Within the field of theoretical computer science, category theory is able to provide real benefits. The case of term unification is a good example of this: First, the correctness proof of the traditional unification algorithm may be greatly simplified when approached from a categorical point of view. Second, the notion of what unification can be broadened in an elegant manner from the simple term unification to also be applicable on differential equations and type inference [67]. In his influential article *A Categorical Manifesto*, Goguen [68] presents a good overview of the potential benefits – and pitfalls – of applying category theory within the field of theoretical computer science.

The notions of category theory have not only been applied in the strictly theoretical area of computer science, though, it has

also proven itself quite useful in practical applications. As an example of this, one might mention the elegant way in which the idea of *monads* has allowed pure functional languages, such as Haskell, to encapsulate and separate effect-full computations without compromising purity [108, 135].

We will in this monograph makes use of a number of well known algebraic and category theoretical structures. These will be briefly, but relatively thoroughly, recollected here to establish the vocabulary and notational framework used in the remaining text. Extra effort will be spend on the aforementioned monads as they, and their generalizations, will play a crucial part in the developments to follow. The various definitions and notations found here are adapted primarily from the well known books by Adámek et al. [2]; Awodey [8]; Barr and Wells [14]; as well as the eminent book of MacLane [101].

2.1 ALGEBRA PRELIMINARIES

Before proceeding to describe categorical constructions it is well advised to recall some essential notions from universal algebra. In particular, we will here somewhat informally define a few algebraic structures that will prove crucial in the constructions to follow. For a rigorous treatment of universal algebra, please see, e. g., Burris and Sankappanavar [25] or the original and influential work of Birkhoff [20].

Let $\Omega = (n_i)_{i \in I}$ be a family of natural numbers with I being some index set. Then we define an Ω-algebra $\mathfrak{A}$ as a pair (A, F) with A being a non-empty set and $F = (f_i \colon A^{n_i} \longrightarrow A)_{i \in I}$ being a family of functions called *operations*. For some $i \in I$, the operator f_i is said to be n_i-ary but due to their abundance some arities are given special names: A 0-ary operation is called a *constant* or *nullary*, a 1-ary operation is called *unary*. Similarly, 2-ary operations are typically called *binary* and are often written in an infix manner, i. e., if f is a binary operation, then we may write $a \, f \, b$ rather than $f(a, b)$ for $a, b \in A$. Note, while universal algebra in its full generality may consider operations of infinite arity, we will here only consider operations of the finite kind so called

finitary operations. Indeed, nearly no commonly studied algebra actually uses operations of arity greater than two [25].

For simplicity, if the number of operations is finite, e. g., $I = \{1, \ldots, k\}$ for some finite natural number k, then we often write $\mathfrak{A}$ as a tuple $(A, f_1, \ldots, f_k)$. Finally, as will be seen in the following examples, when examining various algebras we often impose additional requirements on the operations using *equational laws*.

The first example of a specific algebra is the simple notion of a *monoid*, this is the name for algebras $\mathfrak{M} = (M, \odot, e)$, with $\odot$ being a binary operation and e being a constant such that $\odot$ is associative and e is an left and right identity element with respect to the $\odot$ operation. That is, the operations satisfy the equational laws

$$\begin{aligned} a \odot (b \odot c) &= (a \odot b) \odot c && \text{Associative law} \\ e \odot a &= a && \text{Left identity law} \\ a \odot e &= a && \text{Right identity law} \end{aligned}$$

for all $a, b, c \in M$. Monoid have many applications but we will introduce it here primarily for being an underlying inspiration for the construction of monoidal categories that will occur later in this chapter.

A *lattice* – for which Birkhoff [22] again provides the premier reference – is an algebra $\mathfrak{L} = (L, \vee, \wedge)$ with $\vee$ and $\wedge$ being binary operations called *join* and *meet*, respectively. Similar to monoids, we impose some restrictions on these operations by means of the following equational laws for all $a, b, c \in L$:

Commutative laws	Associative laws
$a \wedge b = b \wedge a$	$(a \wedge b) \wedge c = a \wedge (b \wedge c)$
$a \vee b = b \vee a$	$(a \vee b) \vee c = a \vee (b \vee c)$

Idempotent laws	Absorption laws
$a \wedge a = a$	$a \wedge (a \vee b) = a$
$a \vee a = a$	$a \vee (a \wedge b) = a$

By removing one or the other operation and keeping the corresponding commutative, associative, and idempotent laws, we

may form the *join semilattice* $(L, \vee)$ and *meet semilattice* $(L, \wedge)$. For a lattice $\mathfrak{L} = (L, \vee, \wedge)$ we can introduce a *partial order* relation, $\leqslant$, by letting

$$a \leqslant b \iff a \vee b = b \quad \text{or}$$
$$a \leqslant b \iff a \wedge b = a$$

for all $a, b \in L$. Clearly by choosing the appropriate definition we may introduce this relation also for semilattices. Note, if $a \leqslant b$ or $b \leqslant a$ holds for all $a, b \in L$ then we call $\leqslant$ a *total order*.

Plain lattices and semilattices come in a great many varieties depending on which additional laws are fulfilled. For example, a lattice $\mathfrak{L} = (L, \vee, \wedge)$ is called *distributive* if the distributive laws

$$a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c) \quad \text{or equivalently}$$
$$a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$$

holds for all $a, b, c \in L$. Similarly, the lattice is called *complete* if all subsets $S \subseteq L$ have both a join and a meet, i. e.,

$$\bigvee S = \bigvee_{s \in S} s \in L \quad \text{and} \quad \bigwedge S = \bigwedge_{s \in S} s \in L.$$

Further, if a semi-lattice only has a join or meet for all non-empty subsets then we say that it is *almost complete*. For a complete lattice $\mathfrak{L} = (L, \vee, \wedge)$ we identity the unique *greatest element* by the $\top$ symbol, i. e., $\top = \bigvee L$ and, dually, the *smallest element* by $\bot = \bigwedge L$. Finally, if $\mathfrak{L}$ is a complete lattice that satisfies the *complete distributive laws* [66]

$$a \vee \bigwedge_{b \in S} b = \bigwedge_{b \in S} (a \vee b) \quad \text{or equivalently}$$
$$a \wedge \bigvee_{b \in S} b = \bigvee_{b \in S} (a \wedge b)$$

for all $a \in L$ and $S \subseteq L$ then we call this lattice a *completely distributive lattice*.

2.2 CATEGORY THEORY PRELIMINARIES

Thus far, a number of algebras have been defined with various properties but the issue of maps from one algebra to another has

been carefully avoided. The purpose of this omission is to first introduce a framework in which such maps may be rigorously treated. This framework is, of course, category theory and we will in this section introduce a number of notions from category theory that will be of use throughout the text.

Definition 2.2.1. A *category* $\mathtt{C} = (\mathrm{O}, \mathrm{Hom}, \circ, \mathrm{id})$ consists of

- a class O of *(*$\mathtt{C}$*-)objects* that we typically denote by $\mathrm{Ob}(\mathtt{C})$;
- for each pair of $\mathtt{C}$-objects A, B, a set $\mathrm{Hom}(A, B)$ of *(*$\mathtt{C}$*-)morphisms*, where $f \in \mathrm{Hom}(A, B)$ typically is written using the lighter notation $f\colon A \longrightarrow B$;
- a composition operator, $\circ$ taking morphisms $f\colon A \longrightarrow B$ and $g\colon B \longrightarrow C$ to a composed morphism $g \circ f\colon A \longrightarrow C$; and
- for each $\mathtt{C}$-object A, an *identity morphism* $\mathrm{id}_A : A \longrightarrow A$.

Finally, to complete the definition, these are such that

(i) composition is associative, that is,

$$h \circ (g \circ f) = (h \circ g) \circ f$$

for morphisms $f\colon A \longrightarrow B$, $g\colon B \longrightarrow C$, and $h\colon C \longrightarrow D$;

(ii) for each morphism $f\colon A \longrightarrow B$, we have

$$f \circ \mathrm{id}_A = f \quad \text{and} \quad \mathrm{id}_B \circ f = f; \text{ and}$$

Note, for a morphism $f\colon A \longrightarrow B$ we say that A is the *domain* of f and B is the *codomain*. Further, to avoid confusion we will sometimes write $\mathrm{Hom}_{\mathtt{C}}(A, B)$ to highlight the underlying category. As an additional remark we should, technically, in the definition also include a condition stating that the domain and codomain of a morphism must be unique, i. e., that the Hom-sets are disjoint. We omit this condition, however, since any structure for which the uniqueness condition does not hold can be trivially altered to a structure for which it does hold. This can be done, e. g., by simply including domain and codomain in the morphism. I. e., for $f \in \mathrm{Hom}(A, B)$ in the non-category we

have $(A, f, B) \in \mathrm{Hom}(A, B)$ in the corresponding category; clearly, composition and identities must be altered accordingly.

With the definition of categories, we can now produce a number of examples of useful categories. In particular, we are now in a position of talking of maps of algebras, or as they are often referred to, *homomorphisms* – which also hints at the origin of the Hom component from the definition above.

Example 2.2.1. Let I be an index set and $\mathfrak{A} = (A, (f_i)_{i\in I})$ and $\mathfrak{B} = (B, (g_i)_{i\in I})$ be two Ω-algebras with $\Omega = (n_i)_{i\in I}$. Then a morphism from $\mathfrak{A}$ to $\mathfrak{B}$ is a function $\alpha\colon A \longrightarrow B$ such that, for each $i \in I$ and elements $a_1, \ldots, a_{n_i} \in A$, we have

$$\alpha(f_i(a_1, \ldots, a_{n_i})) = g_i(\alpha(a_1), \ldots, \alpha(a_{n_i})).$$

Alternatively, we could represent this condition as a *diagram*

$$\begin{array}{ccc} A^{n_i} & \xrightarrow{\alpha^{n_i}} & B^{n_i} \\ {\scriptstyle f_i}\downarrow & & \downarrow{\scriptstyle g_i} \\ A & \xrightarrow[\alpha]{} & B \end{array}$$

which is said to *commute* if the condition holds. The identity homomorphism is simply the identity function from set theory extended to a homomorphism, i. e., for all $i \in I$ we have

$$\mathrm{id}_{\mathfrak{A}}(f_i(a_1, \ldots, a_{n_i})) = f_i(a_1, \ldots, a_{n_i}).$$

Finally, homomorphism composition is just normal function composition that trivially may be shown as associative. For Ω we can then construct a category $\mathtt{Alg}(\Omega)$ have all Ω-algebras as objects and $\mathrm{Hom}(\mathfrak{A}, \mathfrak{B})$ is the set of homomorphisms from the algebra $\mathfrak{A}$ to the algebra $\mathfrak{B}$.

As a specific example, for monoids $\mathfrak{M} = (M, \odot, e)$ and $\mathfrak{N} = (N, \oplus, \iota)$ a monoid-homomorphism from $\mathfrak{M}$ to $\mathfrak{N}$ is a function $\beta\colon M \longrightarrow N$ such that

$$\begin{aligned} \beta(a \odot b) &= \beta(a) \oplus \beta(b) \\ \beta(e) &= \iota. \end{aligned}$$

We will denote the category of all monoids by $\mathtt{Mon}$. The other special cases are the category $\mathtt{Lat}$ of all lattices and the category $\mathtt{ACSLat}$ of almost complete semilattices.

Example 2.2.2. For a category $\mathtt{C}$ we may form the *dual category*, $\mathtt{C}^{\mathrm{op}}$, by inverting the direction of all morphisms of $\mathtt{C}$. Thus, we have $\mathrm{Ob}(\mathtt{C}^{\mathrm{op}}) = \mathrm{Ob}(\mathtt{C})$ and $\mathrm{Hom}_{\mathtt{C}^{\mathrm{op}}}(A, B) = \mathrm{Hom}_{\mathtt{C}}(B, A)$ for all objects A and B. Identities are unchanged but the composition relation, $\circ^{\mathrm{op}}$, in $\mathtt{C}^{\mathrm{op}}$ is defined

$$g \circ^{\mathrm{op}} f = f \circ g.$$

Example 2.2.3. It is worth noting that any monoid, say $\mathfrak{M} = (M, \odot, e)$ is directly interpretable as a category having a single object, say $\bullet$, and $\mathrm{Hom}(\bullet, \bullet) = M$. Composition in the category then corresponds to the monoid operation with the identity morphism being e. In this construction the category laws correspond exactly with the monoid laws and, further, monoid homomorphisms correspond exactly to the forthcoming notion of functors.

Example 2.2.4. The canonical example of a category is $\mathtt{Set}$ having as objects sets and total functions as morphisms.

Example 2.2.5. For some lattice $\mathfrak{L}$, the so-called *Goguen category* $\mathtt{Set}(\mathfrak{L})$ as introduced by Goguen [66] has as objects $\mathfrak{L}$*-fuzzy sets* (A, α) with $A \in \mathrm{Ob}(\mathtt{Set})$ and $\alpha \colon A \longrightarrow L$ a *characteristic function*. Morphisms $f \colon (A, \alpha) \longrightarrow (B, \beta)$ are functions $f \colon A \longrightarrow B$ satisfying the inequality $\alpha \leqslant \beta \circ f$. Composition is normal function composition and the identity morphisms are the regular identity functions. For a point $a \in A$ we typically interpret $\alpha(a)$ to denote the degree of membership of a in the $\mathfrak{L}$-fuzzy set.

It is often useful to assume $\mathfrak{L}$ to be a complete lattice so as to have maximum and minimum values. The same holds for distributivity, and indeed in the rest of this text we typically require $\mathfrak{L}$ to be a completely distributive lattice.

Note, if $L = [0, 1]$ then this construction directly correspond to the original notion of fuzzy sets as given by Zadeh [138]. While Zadeh did consider the general case in this publication, Goguen is generally credited with the development of $\mathfrak{L}$-fuzzy sets.

Example 2.2.6. We may, for two categories $\mathtt{C}$ and $\mathtt{D}$, form the *product category* $\mathtt{C} \times \mathtt{D}$ having as objects all pairs (A, X) with $A \in \mathrm{Ob}(\mathtt{C})$ and $X \in \mathrm{Ob}(\mathtt{D})$ and as morphisms from (A, X) to (B, Y) all pairs (f, g) with $f \in \mathrm{Hom}_{\mathtt{C}}(A, B)$ and $g \in \mathrm{Hom}_{\mathtt{D}}(X, Y)$. The identity morphism for the object (A, X) is simply $(\mathrm{id}_A, \mathrm{id}_X)$ where the component identity morphisms are taken from the corresponding category. Composition of morphisms

$$(f, g)\colon (A, X) \longrightarrow (B, Y) \quad \text{and} \quad (f', g')\colon (B, Y) \longrightarrow (C, Z)$$

is defined pointwise, i. e.,

$$(f', g') \circ (f, g) = (f' \circ f, g' \circ g).$$

The generalization to product categories of finite length, that is, $\mathtt{C}_1 \times \cdots \times \mathtt{C}_n$ for categories $\mathtt{C}_1, \ldots, \mathtt{C}_n$ is straightforward using an inductive argument. We will in Section 2.4 see a further generalization of this construction that allows arbitrary product categories over an index set.

When morphisms satisfy some special criterion we sometimes refer to them using some characteristic names. The already mentioned identity morphisms are an example of this. We will also have occasion to use the term *isomorphism*, referring to a morphism $f\colon A \longrightarrow B$ for which there exists a morphism $g\colon B \longrightarrow A$, called the *inverse of* f, such that

$$g \circ f = \mathrm{id}_A \quad \text{and} \quad f \circ g = \mathrm{id}_B\,.$$

The inverse of an isomorphism f is unique and we often denote this inverse by f^{-1}. We occasionally will use the special arrow $\xrightarrow{\sim}$ to indicate when a morphism is an isomorphism, e. g., $f\colon A \xrightarrow{\sim} B$. Similarly, if there exists an isomorphism between the objects A and B then we write $A \cong B$ and say that they are *isomorphic*.

It is clear from the constructions above that categories place a great deal of emphasis on the notion of maps between structures. It should therefore not come as a surprise that the following definition of maps between categories is an important one.

Definition 2.2.2. A *functor* $\mathsf{F}\colon \mathtt{C} \longrightarrow \mathtt{D}$ taking a category $\mathtt{C}$ to a category $\mathtt{D}$ maps each $\mathtt{C}$-object A to a $\mathtt{D}$-object $\mathsf{F}A$ and each $\mathtt{C}$-morphism $f\colon A \longrightarrow B$ to a $\mathtt{D}$-morphism $\mathsf{F}f\colon \mathsf{F}A \longrightarrow \mathsf{F}B$ such that identities and compositions are preserved, i. e.,

(i) if A is a $\mathtt{C}$-object, then $\mathsf{F}(\mathrm{id}_A) = \mathrm{id}_{\mathsf{F}A}$; and

(ii) if $g \circ f$ is a composite $\mathtt{C}$-morphism, then $\mathsf{F}(g \circ f) = \mathsf{F}g \circ \mathsf{F}f$.

Example 2.2.7. The identity functor for a category $\mathtt{C}$ is denoted $\mathrm{id}_{\mathtt{C}}\colon \mathtt{C} \longrightarrow \mathtt{C}$ and simply maps all objects and morphisms back on themselves. I. e., $\mathrm{id}_{\mathtt{C}}(A) = A$ and $\mathrm{id}_{\mathtt{C}}(f) = f$ for all objects A and morphisms f.

Example 2.2.8. For categories $\mathtt{C}$ and $\mathtt{D}$ and a given $\mathtt{D}$-object A, the *constant object functor* $A_{\mathtt{C}}\colon \mathtt{C} \longrightarrow \mathtt{D}$ assigns all $\mathtt{C}$-objects to A and all morphisms in $\mathtt{C}$ to the identity morphism id_A.

Example 2.2.9. For any category $\mathtt{C}$ and $\mathtt{C}$-object A, we may construct the *covariant* Hom*-functor*, $\mathsf{Hom}(A, -)\colon \mathtt{C} \longrightarrow \mathtt{Set}$, that maps $\mathtt{C}$-morphisms $f\colon B \longrightarrow C$ according to

$$\mathsf{Hom}(A, -)(f\colon B \longrightarrow C) = \mathsf{Hom}(A, f)\colon \mathrm{Hom}(A, B) \longrightarrow \mathrm{Hom}(A, C).$$

where $\mathsf{Hom}(A, f)(g) = f \circ g$.

The notion of composite functors follows from viewing composing the object and morphism mappings separately, i. e., for functors $\mathsf{F}\colon \mathtt{C} \longrightarrow \mathtt{D}$ and $\mathsf{G}\colon \mathtt{D} \longrightarrow \mathtt{E}$ we let define the composite functor $\mathsf{G} \circ \mathsf{F}\colon \mathtt{C} \longrightarrow \mathtt{E}$ by $(\mathsf{G} \circ \mathsf{F})(A) = \mathsf{G}(\mathsf{F}A)$ for all $\mathtt{C}$-objects A and $(\mathsf{G} \circ \mathsf{F})(f) = \mathsf{G}(\mathsf{F}(f))$ for all $\mathtt{C}$-morphisms f. It is straightforward to show that this construction is associative and that the identity functor above indeed act as an identity with respect to the composition. Note, for functor composition we often omit the $\circ$ and simply write GF instead of $\mathsf{G} \circ \mathsf{F}$.

With the construction of functors it is tempting to construct a category of all categories having functors as morphisms. This is, however, problematic in the sense that it leads to paradoxes of the same nature that occurs when attempting to construct the set

of all sets. But, similar to the way there exists a class of all sets we may construct a notion of categories called quasicategories that live in a higher set theoretic universe. This will here not be done formally but we will assume the existence of the quasicategory $\mathtt{Cat}$ having as objects all categories and as morphisms the functors between them.

The final fundamental construction in category theory that will be introduced is the following notion of maps between functors.

Definition 2.2.3. For functors $F, G \colon C \longrightarrow D$, a *natural transformation* $\alpha \colon F \longrightarrow G$ is a family of morphisms $\alpha_A : FA \longrightarrow GA$ for all C-objects A such that

$$\begin{array}{ccc} FA & \xrightarrow{\alpha_A} & GA \\ {\scriptstyle Ff}\downarrow & & \downarrow{\scriptstyle Gf} \\ FB & \xrightarrow[\alpha_B]{} & GB \end{array}$$

commutes for all C-morphisms $f \colon A \longrightarrow B$, i. e., $Gf \circ \alpha_A = \alpha_B \circ Ff$.

The identity natural transformation for a functor $F \colon C \longrightarrow D$, denoted id_F, simply maps to the identity morphism in the target category, i. e., for each C-object A we have $(\mathrm{id}_F)_A = \mathrm{id}_{FA}$. Further, if $F, G, H \colon C \longrightarrow D$ are functors then for natural transformations $\alpha \colon F \longrightarrow G$ and $\beta \colon G \longrightarrow H$ we may construct the composite natural transformation $\beta \circ \alpha \colon F \longrightarrow H$ defined by $(\beta \circ \alpha)_A = \beta_A \circ \alpha_A$.

Natural transformations also have an additional notion of composition called *horizontal composition* – indeed, the regular sometimes referred to as *vertical composition*. This composition gives for natural transformation $\alpha \colon E \longrightarrow F$ and $\beta \colon G \longrightarrow H$ a natural transformation $\alpha \star \beta \colon EG \longrightarrow FH$ such that the diagram

$$\begin{array}{ccccc} & & FGA & & \\ & {\scriptstyle \alpha_{GA}}\nearrow & & \searrow{\scriptstyle F\beta_A} & \\ EGA & & \xrightarrow{\alpha \star \beta} & & FHA \\ & {\scriptstyle E\beta_A}\searrow & & \nearrow{\scriptstyle \alpha_{HA}} & \\ & & EHA & & \end{array}$$

commutes for all objects A.

With the identity natural transformation and natural transformation composition – for which it is straightforward to associativity – it is now possible to construct the *functor quasicategory* $[\mathtt{C}, \mathtt{D}]$ having as objects functors from $\mathtt{C}$ to $\mathtt{D}$ and as morphisms the natural transformations.

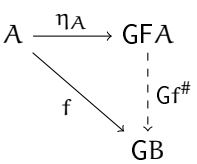
For a natural transformation $\alpha\colon \mathsf{F} \longrightarrow \mathsf{G}$ with $\mathsf{F}, \mathsf{G}\colon \mathtt{C} \longrightarrow \mathtt{D}$ and functor $\mathsf{H}\colon \mathtt{D} \longrightarrow \mathtt{E}$ we introduce the natural transformations

$$\begin{aligned} \mathsf{H}\alpha\colon \mathsf{HF} &\longrightarrow \mathsf{HG} \qquad \text{defined by} \qquad (\mathsf{F}\alpha)_A = \mathsf{F}\alpha_A \\ \alpha\mathsf{H}\colon \mathsf{FH} &\longrightarrow \mathsf{GH} \qquad \text{defined by} \qquad (\alpha\mathsf{F})_A = \alpha_{\mathsf{F}A} \end{aligned}$$

that together provides a convenient short-hand notation when discussing and working with natural transformations. Finally, if a natural transformation $\alpha\colon \mathsf{F} \longrightarrow \mathsf{G}$ is such that α_A satisfy some morphism property 'x' for all objects A, then we say that α is a *natural 'x'* – thus, if each morphism α_A is an isomorphism, then α is said to be a *natural isomorphism*.

One of the major contributions of category theory to the world of mathematics and computer science is the notion of *adjoints* as introduced by Kan [83]. In our case adjoints will appear in the construction of monoidal categories but they are in general a surprisingly powerful tool that arise in many situations and in many fields [121].

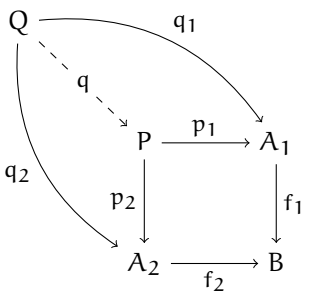
Definition 2.2.4. An *adjunction* is a pair of functors $\mathsf{F}\colon \mathtt{C} \longrightarrow \mathtt{D}$ and $\mathsf{G}\colon \mathtt{D} \longrightarrow \mathtt{C}$ with a natural transformation $\eta\colon \mathrm{id}_{\mathtt{C}} \longrightarrow \mathsf{GF}$ such that for each $\mathtt{C}$-object A and $\mathtt{C}$-morphism $f\colon A \longrightarrow \mathsf{G}B$ there exists a unique morphism $f^{\#}\colon \mathsf{F}A \longrightarrow B$ such that the diagram



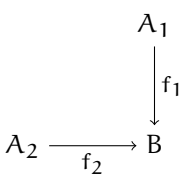
commutes. For this situation we call F a *left adjoint* and G a *right adjoint*.

Another important instrument in category theory are so called universal constructions like limits, e. g., products, pullbacks, and

equalizers, and their duals, colimits, e. g., coproducts, pushouts, and coequalizers. As the name suggests, these are defined universally for all categories and specialize in concrete categories, whenever the construction exist in the concrete category. For illumination, the pullback (also called fibre products) diagram



says that, given the morphisms $f_1 \colon A_1 \longrightarrow B$ and $f_2 \colon A_2 \longrightarrow B$, P, with the morphisms $p_1 \colon P \longrightarrow A_1$ and $p_2 \colon P \longrightarrow A_2$, is pullback, with respect to f_1 and f_2, if, whenever $f_1 \circ q_1 = f_2 \circ q_2$, there exists a unique $q \colon Q \longrightarrow P$, such that the pullback diagram commutes. This is the same as saying that the pullback is the limit to the diagram



that, e. g., in Goguen [69] is seen as a candidate for a general notion of a co-relations in a category. Such relations are closer to relational algebraic views as appearing, e. g., in database theory for queries. Note that this is intuitively an entirely different notion of relations as compared to the view of Kleisli morphisms being general relations.

By way of example, in the case of `Set`, the set

$$P = \{(x_1, x_2) \in A_1 \times A_2 \mid f_1(x_1) = f_2(x_2)\},$$

with $p_1(x_1, x_2) = x_1$ and $p_2(x_1, x_2) = x_2$, is a pullback.

Similarly, the Goguen category $\mathtt{Set}(\mathfrak{L})$, with $\mathfrak{L} = (L, \vee, \wedge)$ being a complete lattice also has pullbacks. In this case we have for (A_1, α_1) and (A_2, α_2) that the $\mathfrak{L}$-fuzzy set (P, β) with

$$\begin{aligned} P &= A_1 \times A_2 \\ \beta(x_1, x_2) &= \alpha_1(x_1) \wedge \alpha_2(x_2) \end{aligned}$$

and morphisms $p_1(x_1, x_2) = x_1$ and $p_2(x_1, x_2) = x_2$ forms a pullback.

2.3 MONADS AND PARTIALLY ORDERED MONADS

Monads were originally recognized and presented, e. g., by Godement [64] and Huber [79]. Since then, monads have been successfully used within algebra, logic and topology, and in a wide range of computer science application, in particular related to language constructions in functional programming. Monads as a useful abstraction in programming and program semantics is well-known and is a central feature of, e. g., the Haskell programming language [108, 109, 135].

Definition 2.3.1. A *monad* $\mathbf{F}$ *over a category* $\mathtt{C}$ is an endofunctor $F\colon \mathtt{C} \longrightarrow \mathtt{C}$ with two associated natural transformations

$$\eta\colon \mathrm{id}_{\mathtt{C}} \longrightarrow F \qquad \text{and} \qquad \mu\colon FF \longrightarrow F,$$

for which

$$\begin{array}{ccc} FFFA & \xrightarrow{\mu_{FA}} & FFA \\ {\scriptstyle F\mu_A}\downarrow & & \downarrow{\scriptstyle \mu_A} \\ FFA & \xrightarrow[\mu_A]{} & FA \end{array} \qquad \text{and} \qquad \begin{array}{ccc} FA & \xrightarrow{\eta_{FA}} & FFA \\ {\scriptstyle F\eta_A}\downarrow & \searrow & \downarrow{\scriptstyle \mu_A} \\ FFA & \xrightarrow[\mu_A]{} & FA \end{array}$$

commutes for all objects A. Note, when it is ambiguous as to which monad η and μ belongs, we indicate their monad by including superscripts. I. e., in these cases we write η^F and μ^F instead of η and μ, respectively.

A large number of example monads may be invoked to give intuition as to their function. We will here only present a few

simple but crucial ones. For more complex monad constructions please see Section 2.7 were a number of *term monads* are defined.

Example 2.3.1. The *covariant powerset monad*, $\mathbf{P} = (P, \eta, \mu)$ is defined

$$\begin{aligned} PX &= \{Y \mid Y \subseteq X\} & Pf(X) &= \{f(x) \mid x \in X\} \\ \eta_X(x) &= \{x\} & \mu_X(X) &= \{x \in Y \mid Y \in X\}. \end{aligned}$$

Example 2.3.2. Very similar to $\mathbf{P}$, we may define the *proper powerset monad* $\mathbf{P}_0 = (P_0, \eta, \mu)$ as follows

$$\begin{aligned} P_0X &= \{Y \mid Y \subseteq X, Y \neq \emptyset\} & P_0f(X) &= \{f(x) \mid x \in X\} \\ \eta_X(x) &= \{x\} & \mu_X(X) &= \{x \in Y \mid Y \in X\}. \end{aligned}$$

It is worth noting that this monad is unique in the sense that it is the only way P_0 can be made a monad [44].

Example 2.3.3. For a completely distributive lattice $\mathfrak{L} = (L, \vee, \wedge)$ we can construct a monad $\mathbf{L} = (L, \eta, \mu)$ over $\mathtt{Set}$ by

$$\begin{aligned} LX &= L^X \\ Lf(A)(y) &= \bigvee_{f(x)=y} A(x) \\ \eta_X(x)(x') &= \begin{cases} \top & \text{if } x = x' \\ \bot & \text{otherwise} \end{cases} \\ \mu_X(M)(x) &= \bigvee_{A \in LX} A(x) \wedge M(A) \end{aligned}$$

Note, with L being a two element set – sometimes such a set is denoted 2 – we have that $\mathbf{L}$ is isomorphic to the regular powerset monad $\mathbf{P}$, we denote this monad $\mathbf{2}$.

Each monad induces – at least – two special categories, the Kleisli category and the Eilenberg–Moore category.

Definition 2.3.2. The *Kleisli category* for a monad $\mathbf{F} = (F, \eta, \mu)$ over $\mathtt{C}$ is a category $\mathtt{C}_\mathbf{F}$ having

$$\mathrm{Ob}(\mathtt{C}^\mathbf{F}) = \mathrm{Ob}(\mathtt{C}) \quad \text{and} \quad \mathrm{Hom}_{\mathtt{C}^\mathbf{F}}(A, B) = \mathrm{Hom}_{\mathtt{C}}(A, FB).$$

The identity morphism for a C_{F}-object A is $\eta_A : A \longrightarrow FA$ and composition of the morphisms $f : A \longrightarrow FB$ and $g : B \longrightarrow FC$ is defined

$$g \circ_{\mathsf{C}^{\mathsf{F}}} f = \mu_z \circ_{\mathsf{C}} Fg \circ_{\mathsf{C}} f$$

When working in the Kleisli category it is often convenient to reverse the order of composition and for that purpose we introduce an operation, $\diamond$, called the *Kleisli composition* defined $f \diamond g = g \circ_{\mathsf{C}^{\mathsf{F}}} f$. Further, for a Kleisli morphism $f : A \longrightarrow FB$ we sometimes use the notation $f : A \dashrightarrow B$.

Indeed, from a computational perspective it is natural to view the Kleisli category as the primitive notion from which monads spring. In particular, inspired by the Kleisli category, we may define a construction equivalent to the monads of Definition 2.3.1.

Definition 2.3.3. A *Kleisli triple* $\mathsf{T} = (T, \eta, (\_)^{\#})$ over a category C consists of a function $T : \mathrm{Ob}(\mathsf{C}) \longrightarrow \mathrm{Ob}(\mathsf{C})$, for each C-object A, a morphism $\eta_A : A \longrightarrow TA$, and an *extension operator*

$$(\_)^{\#} : \mathrm{Hom}(A, TB) \longrightarrow \mathrm{Hom}(TA, TB)$$

for $A, B, C \in \mathrm{Ob}(\mathsf{C})$ such that

$$\begin{aligned} \eta_A^{\#} &= \mathrm{id}_{TA} \\ f^{\#} \circ \eta_A &= f \\ (g^{\#} \circ f)^{\#} &= g^{\#} \circ f^{\#} \end{aligned}$$

for all morphisms $f : A \longrightarrow TB$ and $g : B \longrightarrow TC$.

It is straightforward to show that Kleisli triples and monads are completely equivalent constructions, albeit having different definitions. While we here will employ the monad definition it is worth noting that Kleisli triples often are more convenient and intuitive when applied as a programming pattern that allows a principled management of, e. g., side-effects and partiality in pure functional programming languages [108].

For the powerset monad $\mathbf{P}$ we may make the observation that $\mathsf{Set}^{\mathbf{P}}$ is isomorphic to the category of sets and relations. For this

reason we will here prefer making use of this Kleisli category over any explicit construction of the category of sets and relations. By extension we also prefer to work with the Kleisli category of the fuzzy powerset monad $\mathbf{L}$ instead of working directly with the category of sets and fuzzy relations.

The second important category induced by a monad is the so called Eilenberg–Moore category, first presented in Eilenberg and Moore [42]. From an algebraic perspective, this category essentially shows that monads generalize universal algebra.

Definition 2.3.4. The Eilenberg–Moore category, $\mathtt{C}^{\mathbf{F}}$, for a monad $\mathbf{F} = (F, \eta, \mu)$ over a category $\mathtt{C}$ has as objects $\mathbf{F}$-*algebras*, i. e., pairs (A, h) with $A \in \mathrm{Ob}(\mathtt{C})$ and $h\colon FA \longrightarrow A$ such that the diagrams

$$\begin{array}{ccc} FFA & \xrightarrow{Fh} & FA \\ {\scriptstyle \mu_A}\downarrow & & \downarrow{\scriptstyle h} \\ FA & \xrightarrow[h]{} & A \end{array} \qquad \text{and} \qquad \begin{array}{ccc} A & \xrightarrow{\eta_A} & FA \\ & \searrow & \downarrow{\scriptstyle h} \\ & & A \end{array}$$

commute. A morphism $f\colon (A, h) \longrightarrow (B, g)$ in $\mathtt{C}^{\mathbf{F}}$ is a morphism $f'\colon A \longrightarrow B$ in $\mathtt{C}$ for which the diagram

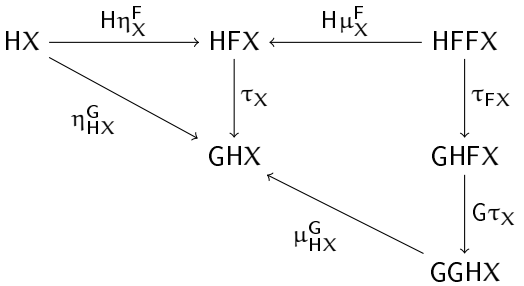
$$\begin{array}{ccc} FA & \xrightarrow{Ff'} & FB \\ {\scriptstyle h}\downarrow & & \downarrow{\scriptstyle g} \\ A & \xrightarrow[f']{} & B \end{array}$$

commutes.

Note, similar to how a Kleisli category may be seen as the source of monads, we may equally well see Eilenberg–Moore categories as the primitive notion. In other words, monads are fully determined by their algebras [101, Theorem VI.1].

Monad morphisms, as given in the following definition, provide us with the means of forming a category of monads, denoted $\mathtt{Mnd}$.

Definition 2.3.5. Let $\mathbf{F} = (F, \eta^F, \mu^F)$ and $\mathbf{G} = (G, \eta^G, \mu^G)$ be two monads over categories $\mathtt{C}$ and $\mathtt{D}$, respectively. Then a morphism from $\mathbf{F}$ to $\mathbf{G}$ is a pair (H, τ) with $H\colon \mathtt{C} \longrightarrow \mathtt{D}$ being a functor and $\tau\colon F \longrightarrow G$ being a natural transformation such that the diagram



commutes.

The monads over a given category $\mathtt{C}$ form a very useful subcategory of $\mathtt{Mnd}$. We call such a subcategory $\mathtt{Mnd}[\mathtt{C}]$ and immediately make the observation that a morphism (H, τ) in $\mathtt{Mnd}[\mathtt{C}]$ will always have $H = \mathrm{id}_{\mathtt{C}}$ and we therefore identify the monad morphisms in $\mathtt{Mnd}[\mathtt{C}]$ by their natural transformation, i. e., we say $\tau\colon \mathbf{F} \longrightarrow \mathbf{G}$ instead of $(\mathrm{id}_{\mathtt{C}}, \tau)\colon \mathbf{F} \longrightarrow \mathbf{G}$.

While the notion of monads introduced above is quite useful indeed we may beneficially enrich monads by adding order structure. This addition will below lead to the notion of *partially ordered monads*. Historically, partially ordered monads are due to Gähler [60, 61] and evolved from studies around filter based convergence structures and Cauchy structures as studied by Kowalsky [91] and Keller [85]. More examples involving the fuzzy filter monad were developed in Eklund and Gähler [44]. Monad and partially ordered monad techniques for compactification were developed in Eklund and Gähler [45], Gähler and Eklund [62], originally inspired by a compactification construction due Richardson [120] for filter based limit spaces. We will in this chapter develop the partially ordered monads in a different direction, namely as a backdrop for developments in Kleene algebra and generalized general logics.

Definition 2.3.6. A *basic triple*, $\Phi = (\mathsf{F}, \leqslant, \eta)$, consists of a functor $\mathsf{F}\colon \mathtt{Set} \longrightarrow \mathtt{Set}$ having an associated natural transformation $\eta\colon \mathrm{id}_{\mathtt{Set}} \longrightarrow \mathsf{F}$ and $\leqslant = (\leqslant_A)_{A \in \mathrm{Ob}(\mathtt{Set})}$ a family of order relations. These are subject to the conditions that for each set A we have that

(i) $(\mathsf{F}A, \leqslant_A)$ is an almost complete semilattice and

(ii) if $A = \emptyset$ then $\mathsf{F}A = \emptyset$.

For simplicity we often leave the subscript of $\leqslant$ and its corresponding binary operation $\vee$ implicit, i. e., writing just $\leqslant$ and $\vee$ instead of $\leqslant_A$ and $\vee_A$.

Note, the original definition in Gähler [60] includes an additional condition stating that for all $x, y \in A$ we have that $\eta_A(x) \vee \eta_A(y)$ exists only if $x = y$. We will here forgo this requirement so as to allow viewing, e. g., the powerset functor as a basic triple. With the condition – which is necessary for topological developments – the best we can do is to view the proper powerset functor as a basic triple. Omitting the condition will cause no problems for the purposes in this monograph.

Given a basic triple $\Phi = (\mathsf{F}, \leqslant, \eta)$, we may construct a functor $(\mathsf{F}, \leqslant)\colon \mathtt{Set} \longrightarrow \mathtt{ACSLat}$ defined such that $A \longmapsto (\mathsf{F}A, \leqslant)$ and it follows immediately from this definition of $(\mathsf{F}, \leqslant)$ that

$$\mathsf{F}f(\bigvee \mathcal{A}) = \bigvee_{A \in \mathcal{A}} \mathsf{F}f(A) \tag{1}$$

for all $\mathtt{Set}$-morphisms $f\colon A \longrightarrow B$ and non-empty $\mathcal{A} \subseteq \mathsf{F}A$. That is, morphisms under F preserve non-empty suprema. Further, if $(\mathsf{F}, \leqslant^{\mathsf{F}}, \eta^{\mathsf{F}})$ and $(\mathsf{G}, \leqslant^{\mathsf{G}}, \eta^{\mathsf{G}})$ are basic triples, then

$$(\mathsf{F} \circ \mathsf{G}, \leqslant^{\mathsf{F}}, \eta^{\mathsf{F}}\mathsf{G} \circ \eta^{\mathsf{G}})$$

is also a basic triple.

Consider L as a functor from $\mathtt{Set}$ to $\mathtt{ACSLat}$ with $\mathcal{A} \leqslant \mathcal{B}$ for all $\mathcal{A}, \mathcal{B} \in \mathsf{L}A$ having the meaning $\mathcal{A}(x) \leqslant \mathcal{B}(x)$ for all $x \in A$. Then $(\mathsf{L}, \leqslant, \eta)$ is a basic triple where $\eta_A : A \longrightarrow \mathsf{L}A$ is as given in Example 2.3.3.

With the basic triples as a base, we may now return to the main focus and define partially ordered monads.

Definition 2.3.7. A *partially ordered monad* $\mathbf{F} = (F, \leqslant, \eta, \mu)$ is such that

(i) $(F, \leqslant, \eta)$ is a basic triple.

(ii) $\mu\colon FF \longrightarrow F$ is a natural transformation making (F, η, μ) a monad.

(iii) For all morphisms $f, g\colon A \longrightarrow FB$, if $f \leqslant g$, i. e., $f(x) \leqslant g(x)$ for all $x \in A$, then

$$\mu_B \circ Ff \leqslant \mu_B \circ Fg.$$

(iv) $\mu_A : (FFA, \leqslant) \longrightarrow (FA, \leqslant)$ preserves non-empty suprema. In other words,

$$\mu_A(\bigvee \mathcal{M}) = \bigvee_{M \in \mathcal{M}} \mu_A(M) \tag{2}$$

for non-empty $\mathcal{M} \subseteq FFA$.

We may immediately observe that (2) together with the known existence of a suprema implies that

$$\mu_B \circ F(\vee_i f_i) = \vee_i(\mu_B \circ Ff_i) \tag{3}$$

given any family of morphisms $\{f_i\colon A \longrightarrow FB\}_{i\in I}$ and where the morphism $\vee_i f_i\colon A \longrightarrow FB$ is defined by $(\vee_i f_i)(x) = \bigvee_{i\in I} f_i(x)$. This holds since $g \leqslant \vee_i f_i$ for all $g \in \{f_i\colon A \longrightarrow FB\}_{i\in I}$ and the resulting inequality $\mu_B \circ Fg \leqslant \mu_B \circ F(\vee_i f_i)$ subsequently give the identity of (3).

Example 2.3.4. The basic triple $(L, \leqslant, \eta^L)$ can be extended to a partially ordered monad [62] using the multiplication μ as given in Example 2.3.3. In the following we illustrate in some detail that this monad have the properties given in (1), (2), and (3).

First, given $f\colon X \longrightarrow Y$ and non-empty $\mathcal{B} \subseteq \mathsf{L}X$ we show (1) by

$$\begin{aligned}
\mathsf{L}f(\bigvee \mathcal{B})(y) &= \mathsf{L}f(\bigvee_{B\in\mathcal{B}} B)(y) \\
&= \bigvee_{f(x)=y} (\bigvee_{B\in\mathcal{B}} B)(x) \\
&= \bigvee_{B\in\mathcal{B}} \bigvee_{f(x)=y} B(x) \\
&= \bigvee_{B\in\mathcal{B}} \mathsf{L}f(B)(y).
\end{aligned}$$

Second, given $\mathcal{M} \subseteq \mathsf{LL}X$ and $x \in X$ we have (2) by

$$\begin{aligned}
\mu_X(\bigvee \mathcal{M})(x) &= \bigvee_{B\in\mathsf{L}X} B(x) \wedge \bigvee_{M\in\mathcal{M}} M(B) \\
&= \bigvee_{B\in\mathsf{L}X} \bigvee_{M\in\mathcal{M}} B(x) \wedge M(B) \\
&= \bigvee_{M\in\mathcal{M}} \bigvee_{B\in\mathsf{L}X} B(x) \wedge M(B) \\
&= \bigvee_{M\in\mathcal{M}} \mu_X(M)(x).
\end{aligned}$$

Finally, for $B \in \mathsf{L}X$ and $x \in X$, (3) is shown by

$$\begin{aligned}
[\mu_X \circ \mathsf{L}(\vee_i f_i)](B)(x) &= \bigvee_{B'\in\mathsf{L}X} B'(x) \wedge \bigvee_{(\vee_i f_i)(y)=B'} B(y) \\
&= \bigvee_{B'\in\mathsf{L}X} \bigvee_{(\vee_i f_i)(y)=B'} [B'(x) \wedge B(y)] \\
&= \bigvee_{y\in Y} [\bigvee_{i\in I} f_i(y)(x) \wedge B(y)] \\
&= \bigvee_{i\in I} \bigvee_{y\in Y} [f_i(y)(x) \wedge B(y)] \\
&= \bigvee_{i\in I} \bigvee_{B'\in\mathsf{L}X} \bigvee_{f_i(y)=B'} [f_i(y)(x) \wedge B(y)] \\
&= \bigvee_{i\in I} (\bigvee_{B'\in\mathsf{L}X} B'(x) \wedge \mathsf{L}f_i(B)(B')) \\
&= (\vee_i(\mu_X \circ \mathsf{L}f_i))(B)(x).
\end{aligned}$$

By defining morphisms of partially ordered morphisms we can extend the previously defined category of monads, `Mnd`, to the category of partially ordered monads, `POMnd`. Intuitively, this morphism construction implicitly encodes the requirement that τ also is a morphism of the monads' underlying basic triples.

Definition 2.3.8. Let $\mathbf{F} = (F, \leqslant^F, \eta^F, \mu^F)$ and $\mathbf{G} = (G, \leqslant^G, \eta^G, \mu^G)$ be two partially ordered monads. Then a morphism from $\mathbf{F}$ to $\mathbf{G}$ is a natural transformation $\tau\colon (F, \leqslant^F) \longrightarrow (G, \leqslant^G)$ such that $\tau\colon F \longrightarrow G$ is a monad morphism.

It is from this definition immediate that the identity monad morphism is identified with the identity natural transformation. Further, it is easy to see that the category conditions are satisfied and `POMnd` truly forms a category of partially ordered monads. Note, `POMnd` is a subcategory of `Mnd`[`Set`].

2.4 SORTED CATEGORIES

In the one-sorted (and crisp) case for signatures we typically work in `Set`, but in the many-sorted (and crisp) case we need the "sorted category of sets" for the many-sorted term functor. We start this section by a more general view by considering "a sorted category of objects". This need leads to the idea of a product category over some, possibly infinite, index set. While the construction is generally applicable for any index set, we will here focus entirely on index sets representing *sorts* (or types) and develop the vocabulary accordingly.

Let S be an index set (in ZFC) whose indices are called sorts, and we do not assume any order on S. For a category `C`, we write $\mathtt{C}_S$ for the product category $\prod_S \mathtt{C}$. The objects of $\mathtt{C}_S$ are families $(X_s)_{s\in S}$ such that $X_s \in \mathrm{Ob}(\mathtt{C})$ for all $s \in S$. We will also use X_S as a shorthand notation for these families. The morphisms between objects $(X_s)_{s\in S}$ and $(Y_s)_{s\in S}$ are families $(f_s)_{s\in S}$ such that $f_s \in \mathrm{Hom}_{\mathtt{C}}(X_s, Y_s)$ for all $s \in S$, and similarly we will use f_S as a shorthand notation. The composition of morphisms is defined sortwise (componentwise), i. e., $(g_s)_{s\in S} \circ (f_s)_{s\in S} = (g_s \circ f_s)_{s\in S}$. Finally, the identity map id_{X_S} for an object X_S in $\mathtt{C}_S$ is simply the family of identities $(\mathrm{id}_{X_s})_{s\in S}$.

Functors $F_s\colon C \longrightarrow D$ are lifted to functors $F = (F_s)_{s\in S}$ from C_S to D_S, so that, e. g., the regular powerset functor $P_S = (P)_{s\in S}$ and the regular many-valued powerset functor $L_S = (L)_{s\in S}$, both are lifted to functors on $\mathtt{Set}_S$.

Products and coproducts, $\prod$ and $\coprod$, are handled sortwise. We also have a "subobject relation", thus, $(X_s)_{s\in S} \subseteq (Y_s)_{s\in S}$ if and only if $X_s \subseteq Y_s$ for all $s \in S$. It is clear that all limits and colimits exist in $\mathtt{Set}_S$, because operations on $\mathtt{Set}_S$-objects are defined sortwise for sets. Further, the product $\prod_{i\in I} F_i$ and coproduct $\coprod_{i\in I} F_i$ of covariant functors F_i over $\mathtt{Set}_S$ are defined as

$$(\prod_{i\in I} F_i)(X_s)_{s\in S} = \prod_{i\in I} F_i(X_s)_{s\in S}$$

and

$$(\coprod_{i\in I} F_i)(X_s)_{s\in S} = \coprod_{i\in I} F_i(X_s)_{s\in S}$$

with morphisms being handled accordingly.

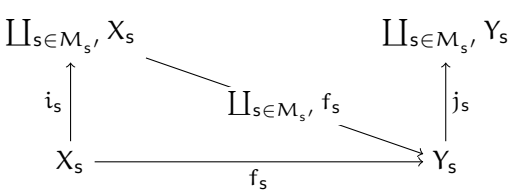
If C has all coproducts and the unique coproduct morphism of a family of identities is a section of the canonical coproduct injection then it is worth noting that a function $\mathfrak{s}\colon S \longrightarrow S'$ give rise to a functor $\mathfrak{S}\colon C_S \longrightarrow C_{S'}$. In this construction we consider sets $M_{s'}$ such that

$$M_{s'} = \{s \in S \mid s' = \mathfrak{s}(s)\}$$

for each $s' \in S'$. Then $\mathfrak{S}$ maps objects $(X_s)_{s\in S}$ to objects $(X_{s'})_{s'\in S'}$ where

$$X_{s'} = \coprod_{s\in M_{s'}} X_s.$$

The mapping of morphisms $(f_s)_{s\in S}\colon (X_s)_{s\in S} \longrightarrow (Y_s)_{s\in S}$ follows immediately from the commutative diagram of coproducts, i. e.,



with i_s, j_s being the canonical coproduct injections. From this we see that $(f_s)_{s\in S}\colon (X_s)_{s\in S} \longrightarrow (Y_s)_{s\in S}$ maps to

$$i_s \circ \coprod_{s\in M_{s'}} f_s \colon \coprod_{s\in M_{s'}} X_s \longrightarrow \coprod_{s\in M_{s'}} Y_s,$$

giving

$$\Big(i_s \circ \coprod_{s\in M_{s'}} f_s\Big)_{s'\in S'} \colon (X_{s'})_{s'\in S'} \longrightarrow (Y_{s'})_{s'\in S'}.$$

Note, for an identity morphism in $\mathtt{C}_S$, e. g., $(\mathrm{id}_{X_s})_{s\in S}$ we have

$$\mathrm{id}_{X_s} \longmapsto i_s \circ \coprod_{s\in M_{s'}} \mathrm{id}_{X_s}$$

for each $s \in S$. Now, let

$$\mathrm{id}^{s'} \colon \coprod_{s\in M_{s'}} X_s \longrightarrow \coprod_{s\in M_{s'}} X_s$$

denote the identity morphism of $\coprod_{s\in M_{s'}} X_s$ and $\coprod \mathrm{id}$ denote the unique coproduct morphism of $(\mathrm{id}_{X_s})_{s\in M_{s'}}$. For each s, we then have

$$\coprod \mathrm{id} \circ \mathrm{id}^{s'} = \mathrm{id}_{X_s} \circ \coprod \mathrm{id} = \coprod \mathrm{id} \circ i_s \circ \coprod \mathrm{id}$$

Finally, $\coprod \mathrm{id}$ is a monomorphism by virtue of being a section [2, Proposition 7.35] and it is therefore left-cancellable. This now gives

$$(\mathrm{id}^{s'})_{s'\in S'} = \mathfrak{S}((\mathrm{id}_{X_s})_{s\in S})$$

and $\mathfrak{S}$ therefore maps identities to identities. To establish $\mathfrak{S}$ as a functor it now suffices to simply note that $\mathfrak{S}$ preserves composition by virtue of existence of compositions in the underlying coproduct diagram. Both $\mathtt{Set}$ and $\mathtt{Set}(\mathfrak{L})$ have all coproducts and for a family of identities, their coproduct morphism is a section. The above functor construction is consequently applicable for both of these categories.

Note, the heavy reliance on coproducts in the construction of the $\mathfrak{S}$ functor stems from the necessity of distinguishing similarly 'named' variables in when interpreting objects of a sorted category as carriers of variables in the upcoming term construction. For example, if a map $\mathfrak{s}\colon S \longrightarrow S'$ is such that $\mathfrak{s}(s) = \mathfrak{s}(s')$ then $\mathfrak{S}$ must ensure that a variable x of sort s is distinct from a variable x of sort s'; failure to do so may compromise the soundness of the logic we wish to construct. Traditionally, this is accomplished by stipulating that a family of variables X_S is pairwise disjoint but the present method is more flexible and generally applicable over a wider variety of underlying categories.

The category $\mathtt{Set}(\mathfrak{L})_S$ is called the *many-sorted Goguen category*. Objects in this category are families of pairs $((X_s, \alpha_s))_{s\in S}$ as objects, where for each $s \in S$, $\alpha_s\colon X_s \longrightarrow L$ is a function (in ZFC). So, fixing $s \in S$ we consider pairs (X_s, α_s) as objects in $\mathtt{Set}(\mathfrak{L})$. Now, the $\mathtt{Set}(\mathfrak{L})$-morphisms $f_s\colon (X_s, \alpha_s) \longrightarrow (Y_s, \beta_s)$ form morphisms

$$(f_s)_{s\in S}\colon ((X_s, \alpha_s))_{s\in S} \longrightarrow ((Y_s, \beta_s))_{s\in S}.$$

It is known that all limits and colimits exist in $\mathtt{Set}(\mathfrak{L})$. For example, we have the *inductive limit* – also known under the name *direct colimit* – of the inductive system

$$(((X_s, \alpha_s))^{(i)}_{s\in S})_{i\in\mathbb{N}} = (((X^{(i)}_s, \alpha^{(i)}_s))_{s\in S})_{i\in\mathbb{N}},$$

written as

$$\operatorname{ind}\varinjlim ((X^{(i)}_s, \alpha^{(i)}_s))_{s\in S}$$

being an object $((Y_s, \beta_s))_{s\in S}$ in $\mathtt{Set}(\mathfrak{L})_S$. Here, $Y_s = \bigcup_{i\in\mathbb{N}} X^{(i)}_s$, and β_s is defined as follows. For $x \in Y_s$, we have

$$A = \{i \in \mathbb{N} \mid x \in X^{(i)}_s\},$$

and we can define

$$\beta_s(x) = \bigvee\{\alpha^{(i)}_s(x) \mid i \in A\}.$$

2.5 MONOIDAL BICLOSED CATEGORIES

The idea of constructing monoidal categories arise from the observation that categorical product, such as cartesian product in

Set, is in certain situation a too strong and restrictive construction. Due to this, given a category C, the notion of product can be weakened by instead considering a bifunctor $\otimes\colon \mathtt{C} \times \mathtt{C} \longrightarrow \mathtt{C}$ that usually is called *tensor product*. It is, similar to cartesian products, a custom to write $A \otimes B$ instead of $\otimes(A, B)$, for objects A and B in $\mathrm{Ob}(\mathtt{C})$.

If the bifunctor also give rise to adjunctions we come to the issue of so called *monoidal closedness* and *monoidal closed categories*. In particular, such adjunctions also give rise to a natural isomorphism that closely mirror the notion of *currying*, as known in type theory. There is indeed the intuition that the monoidal product acts like a composition, i. e., strengthening the currying view of this natural isomorphism. Further, the natural isomorphism also invites to thinking about morphisms carrying uncertainty, and this will open up considerations even for sorts being uncertain, as we more clearly distinguish the name of the sort from how it appears as a morphism.

The branch of category theory exploring the properties of these and similar constructions is usually called *categorical algebra*. Parenthetically, the history of monoidal closed categories, and categorical algebra more broadly, goes back to the study of such natural equivalences[1] by Eilenberg and MacLane [41], in turn inspired, e. g., by theories of linear operators [10] and homology theory. Incidentally, from fuzzy arithmetic point of view it is very much unknown how natural equivalences relate to the generalized arithmetic of Birkhoff [21], on bridging the gap between cardinal and ordinal arithmetic using a partially ordered set view of numbers, as the construction of "large numbers of functors" including natural equivalences and transformations between these functors [41]. Thereafter, the notion of *adjoint functors* [83] came to play an important role for the bifunctor, and for monoidal closed categories to be more formally defined in MacLane [99] and analyzed with respect to the *coherence condition* [87]. In the beginning, monoidal categories were called *categories with multiplication* by Bénabou [17, 18] and MacLane [100]. The name *monoidal closed category* emerges more or less in Eilenberg and Kelly

1 Natural equivalence is simply what we call natural isomorphism.

[40], and attains its simple and clean formulation in MacLane [101].

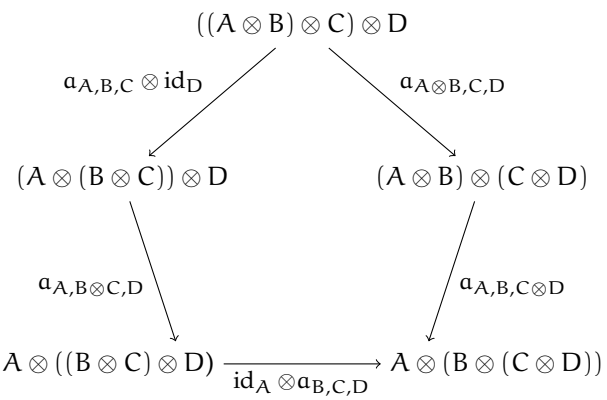
More formally, the definition of monoidal closed categories, following the notational style of Kelly [88], may be given as follows.

Definition 2.5.1. A category $\mathtt{C}$ is said to be a monoidal category if it is equipped with

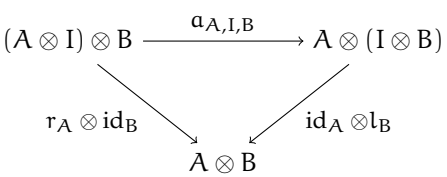
- a bifunctor $\otimes\colon \mathtt{C} \times \mathtt{C} \longrightarrow \mathtt{C}$, called the *tensor product*;
- $I \in \mathrm{Ob}(\mathtt{C})$ a *unit object*; and
- three natural isomorphisms

$$\begin{aligned} a_{A,B,C} &\colon (A \otimes B) \otimes C \xrightarrow{\sim} A \otimes (B \otimes C), \\ l_A &\colon I \otimes A \xrightarrow{\sim} A, \text{ and} \\ r_A &\colon A \otimes I \xrightarrow{\sim} A \end{aligned}$$

such that the diagrams



and



commute.

Note, for the unit object I we always have $r_I = l_I$ [82, Proposition 1.1]. Additionally, a monoidal category becomes a *monoidal (left) closed category*, if the functor $- \otimes B \colon \mathtt{C} \longrightarrow \mathtt{C}$ has a right adjoint, denoted $[B, -]$, for all objects B. Correspondingly, it is right closed if $A \otimes - \colon \mathtt{C} \longrightarrow \mathtt{C}$, for all objects A, has a right adjoint. A monoidal closed category is *biclosed*, if it is both left and right closed, and a *symmetric monoidal category*, whenever the tensor product is commutative. All monoidal closed category that is symmetric is biclosed but a symmetric monoidal category is not necessarily monoidal closed [88, p. 14]. Note that, in this context, $\mathrm{Hom}(I, -)$ plays the role of the *basic functor* [40] up to natural isomorphism since

$$\mathrm{Hom}(A, B) \cong \mathrm{Hom}(I, [A, B]).$$

As an example, the tensor product for vector spaces is monoidal closed, but not cartesian closed. An example more related to our purposes is that given an underlying completely distributive lattice, $\mathfrak{L}$, the Goguen category $\mathtt{Set}(\mathfrak{L})$ is, as shown by Höhle and Stout [78, Theorem 5.3.2] and Stout [131, Theorem 1], a symmetric monoidal closed category and it is therefore also biclosed. In fact, this holds not only for completely distributive lattices but also for the more general concept of so called *commutative and unital quantales*. More concretely, for a completely distributive lattice $\mathfrak{L} = (L, \vee, \wedge)$ we find that $\mathtt{Set}(\mathfrak{L})$ is a monoidal biclosed category if we let

$$\begin{aligned} I &= (\{\emptyset\}, \top) \\ (A, \alpha) \otimes (B, \beta) &= (A \times B, \alpha \wedge \beta) \\ a_{A,B,C}(((x, y), z), \alpha) &= ((x, (y, z)), \alpha) \\ l_A((\{\emptyset\}, x), \alpha) &= (x, \alpha) \\ r_A((x, \{\emptyset\}), \alpha) &= (x, \alpha). \end{aligned}$$

Note that having $\{\emptyset\}$ in I is an arbitrary choice; it may equally well have been any terminal object in $\mathtt{Set}$, i. e., any singleton set.

Adding further structure to $\mathrm{Hom}(A, B)$ is sometimes desirable, e. g., by having operations on morphisms satisfying certain properties. While not explicitly covered more here, this idea brings us

to 2-categories, and from there onward. Morphisms carrying uncertainties would also mean adding structure to the Hom-sets. In specific situations, the Hom-sets may indeed already carry some algebraic or topological structure, and this is then explored further, as required by the context. The name *enriched category theory*, a topic explored in depth by Kelly [88], indeed comes from this enrichment of Hom-sets, or replacing them with monoidal objects constructed using the bifunctor.

Generally speaking it is important to say that uncertainty and fuzziness is enrichness to, not generalization of, the crisp world. In this context one may consider a basic functor, called V by Eilenberg and Kelly [40, p. 421], that removes the enrichment from Hom-objects and therefore plays the important role of a 'defuzzifier'.

2.6 SIGNATURES

Traditional universal algebra, as seen in Section 2.1, primarily considers operations acting only on one *sort* of data. For example, one may define the monoid instance of natural numbers with regular multiplication as the binary operation and 1 as the identity constant; in this case the sort of data may be denoted by some syntactic symbol `nat` that in the algebra may be mapped to the set $\mathbb{N}$. The obvious generalization – as explored by, e. g., Bénabou [19] – is then to consider many-sorted algebras and corresponding many-sorted terms. We will in this section present a notion of *signatures* that is sufficiently general to describe many-sorted operators with possible additional structure to provide non-classical extensions such as fuzzy operators. We will from here on distinguish between 'operation' and 'operator', in particular, we will use 'operator' to mean a syntactic symbol that may be interpreted using a semantic 'operation'. This difference will later be made formal.

While computer scientists frequently speak of *types* instead of sorts but we will in this text follow the algebraic tradition of saying sorts. Further, mathematicians have been influenced, e. g., by Bénabou [19] where the sequence of the argument sorts is seen as an element in the free monoid over a set of sorts.

It is important to note that the shift from one-sorted signatures to many-sorted signatures is often seen as a mere technicality, but very few actually bother to check detail. It is, however, fairly well-known that the shift to many-sortedness is not always that evident and indeed – since we in our case wish to construct a term functor over richer underlying categories than $\mathtt{Set}$ – the many-sorted cases are far from trivial, as we will see.

In the traditional notation of computer science, a many-sorted signature $\Sigma = (S, \Omega)$ consists of a set S of sorts, and a set Ω of operators. The question as to which scope and what extend a signature is a categorical object is non-trivial, and the categorization can be arranged in different ways, as seen below. The intuitive view in computer science is that S is an index set, and reside in ZFC, whereas Ω may be an object in $\mathtt{Set}_S$. Thus, $\Omega = (\Omega_s)_{s \in S}$, where Ω_s is an object in $\mathtt{Set}$, and we would say informally that $\Sigma = (S, \Omega)$ is a *signature over* $\mathtt{Set}$. Operators in Ω_s are then syntactically written as $\omega \colon s_1 \times \cdots \times s_n \to s$, and we say that the arity of $\omega \colon s_1 \times \cdots \times s_n \to s$ is n, or that it is an n-ary operator. The 0-ary operators $\omega \colon \to s$ are called constant operators, or constants. For simplicity, when we will emphasize the sorts or arities in syntactic expressions we write $\omega \colon s_1 \times \cdots \times s_n \to s$, and in the case that sorts and arities are known we may write only ω for the operators also in syntactic expressions. Note that each set Ω_s does not make preferences for arities, it only states that "the end sort" of the operator, or "the result sort" of the operator, is always s. Notice also that $\times$ and $\to$ in the syntactic notation for operators at this point come without any meaning, the interpretation of the signature in an algebras will eventually provide $\times$ and $\to$ with meaning in the underlying category.

In the many-sorted term construction it will be convenient to use the notation $\Omega^{s_1 \times \cdots \times s_n \to s}$ for the set, as an object in $\mathtt{Set}$, of operators $\omega \colon s_1 \times \cdots \times s_n \to s \in \Omega_s$ with n given, and $\Omega^{\to s}$ for the set of constants $\omega \colon \to s$. With these notations we keep explicit track of operator sorts as well as their arities and we assume the identity

$$\Omega_s = \coprod_{\substack{s_1, \ldots, s_n \\ n \leqslant k}} \Omega^{s_1 \times \cdots \times s_n \to s}$$

for some finite $k \in \mathbb{N}$. For example, if $S = \{s, t\}$ then a set $\Omega^{s\times t\to s}$ may differ from $\Omega^{t\times s\to s}$. Clearly, if S is one-pointed, then Σ is a one-sorted signature.

If $\Sigma_1 = (S_1, \Omega_1)$ and $\Sigma_2 = (S_2, \Omega_2)$ are signatures, then a signature mapping $\sigma\colon \Sigma_1 \longrightarrow \Sigma_2$ is a pair of mappings

$$(\mathfrak{s}\colon S_1 \longrightarrow S_2, \mathfrak{o}\colon \Omega_1 \longrightarrow \Omega_2)$$

such that for each Σ_1-operator

$$\omega_1 \in \Omega_1^{s_1\times\cdots\times s_n\to s},$$

there exists an Σ_2-operator

$$\omega_2 \in \Omega_2^{\mathfrak{s}(s_1)\times\cdots\times\mathfrak{s}(s_n)\to\mathfrak{s}(s)}$$

such that $\mathfrak{o}(\omega_1) = \omega_2$. For signature mappings

$$\sigma = (\mathfrak{s}, \mathfrak{o})\colon \Sigma_1 \longrightarrow \Sigma_2 \quad \text{and} \quad \rho = (\mathfrak{r}, \mathfrak{n})\colon \Sigma_2 \longrightarrow \Sigma_3,$$

the composite mapping $\rho \circ \sigma\colon \Sigma_1 \longrightarrow \Sigma_3$ is simply defined pairwise by $(\mathfrak{r}\circ\mathfrak{s}, \mathfrak{n}\circ\mathfrak{o})$. Composition of signature mappings is associative, and we may then anticipate the formal definition of a category of signatures.

Following a strictly mathematical view, in Bénabou [19] signatures are called S-*schéma* and act over sets of sorts. Such signatures consist of triples (Ω, τ, S) where Ω is seen only as a set of operator symbols, $\tau\colon \Omega \longrightarrow \hat{S} \times S$ attaches sorts to these operator symbols – here the set of sorts S is extended to the free monoid $\mathfrak{S} = (\hat{S}, \cdot, e)$ so that elements in $\hat{S}$ reflect sequences of types. Using the computer science notation, an operator $\omega \in \Omega^{s_1\times\cdots\times s_n\to s}$ would have $\tau(\omega) = (s_1 \cdots s_n, s)$. Note also how the monoidal extension, $\hat{\mathfrak{s}}$, of the signature mapping $\mathfrak{s}$ will have the monoid homomorphism property.

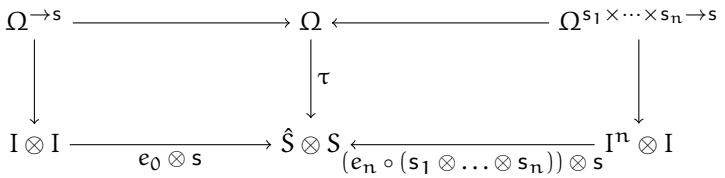
Recall that we from the point of view of computer science see S simply as a set of sorts, i. e., mostly as an index set in ZFC and not as an object in the category `Set`. To contrast, in the above monoidal view of sequences of sorts, S is an object of `Set` and $\mathfrak{S}$ an object on `Mon`, the category of monoids. This change of viewpoint allow us to formally speak of *signatures over* `Set`.

The category $\mathtt{Set}$ is monoidal biclosed with respect to the cartesian product, and signatures can more generally be arranged over any monoidal biclosed category $\mathtt{C}$ with tensor product $\otimes$. Indeed, let S be an object of $\mathtt{C}$, i.e., S represents the "sort set" in a generalized sense. If we by S^0 denote the unit object I in $\mathtt{C}$, we can write $S^{n+1} = S \otimes S^n$, $n \in \mathbb{N}$, and define $\hat{S} = \coprod_{n\in\mathbb{N}} S^n$. Sorts $\mathtt{s}$ can now be recovered in S as morphisms $\mathtt{s}\colon I \longrightarrow S$, and *one-sortedness* would mean that S is a unit object, which in $\mathtt{C}$ means we require that S is a terminal object in $\mathtt{C}$.

With $\mathtt{Set}$ as the underlying category for the monoidal biclosed category with the cartesian product as the tensor product, the unit object in the monoidal biclosed category is a terminal object in $\mathtt{Set}$. Note that every one-pointed set is a terminal object in $\mathtt{Set}$, so the unit object is some selected terminal object, e.g., represented by the one-pointed set $\{\emptyset\}$. Note also that the arrow $\mathtt{s}\colon I \longrightarrow S$ can be identified with an element in S, and vice verse, any element in S generates such a morphism. This means that there is a bijection between S and $\mathrm{Hom}_{\mathtt{Set}}(I, S)$. Unfortunately, as will be shown below, in the more general cases, the objects S and $\mathrm{Hom}_{\mathtt{C}}(I, S)$ are not necessarily isomorphic. For this situation $\mathrm{Hom}_{\mathtt{C}}(I, S)$ represents the sorts, whereas S acts only as a 'base' for that construction of that set of sorts. For simplicity we often introduce some notation, e.g., $\mathtt{S}$ to stand for the set $\mathrm{Hom}_{\mathtt{C}}(I, S)$ of sorts, so that we can write $\mathtt{s} \in \mathtt{S}$ which then intuitively means $\mathtt{s}$ *is a sort in* S in the traditional sense on computer science oriented reading and informal notation.

As we noted in Section 2.5, if $\mathfrak{L}$ is a completely distributive lattice, then $\mathtt{Set}(\mathfrak{L})$ is a monoidal biclosed category, where again there are many choices of unit objects but we keep as unit object $I = (\{\emptyset\}, \top)$. In this case we again have that there is a one-to-one correspondence between S and $\mathrm{Hom}_{\mathtt{Set}(\mathfrak{L})}(I, (S, \top))$ but this is not necessarily the case for $\mathrm{Hom}_{\mathtt{Set}(\mathfrak{L})}(I, (S, \alpha))$, with α being some general characteristic function. Note that it is intuitively desirable indeed to have $(S, \top)$ correspond to "crisp sorts".

Given the sorts, we can now construct the "operator sets" using the pullback squares



where e_n are the canonical morphisms of the coproduct, with e_0 mapping to the unit in $\hat{S}$. We then say that (Ω, τ, S) is a *signature over* $\mathtt{C}$, or a $\mathtt{C}$*-signature*.

This intuitively also relates to the situation where the inverse image of a set with respect to a function can be viewed as a pullback for the corresponding inclusion map and the function. Indeed, in this intuition, $\Omega^{\to s}$ is kind of the inverse image of $I \otimes I$, and similarly $\Omega^{s_1 \times \cdots \times s_n \to s}$ of $I^n \otimes I$ with respect to τ.

Example 2.6.1. In order to verify that $\Omega^{\to s}$ and $\Omega^{s_1 \times \cdots \times s_n \to s}$ really corresponds to our intended objects of operators, let us consider the situation of $\mathtt{C} = \mathtt{Set}$. We will first look at $\Omega^{\to s}$. By the definition of pullback in $\mathtt{Set}$ we have that $\Omega^{\to s}$ is isomorphic to the set

$$\{(\omega, (\emptyset, \emptyset)) \in \Omega \times (I \times I) \mid \tau(\omega) = (e_0, s)(\emptyset, \emptyset)\}.$$

Note that $e_0(\emptyset) = e$, i. e., the "empty string", and that the pullback requirement

$$\tau \circ p_1 = (e_0, s) \circ p_2$$

where $p_1(\omega, (\emptyset, \emptyset)) = \omega$ and $p_2(\omega, (\emptyset, \emptyset)) = (\emptyset, \emptyset)$ then means that $\tau(\omega) = (e, s)$, i. e., ω is of the form $\omega\colon \ \to s$, as expected. By the same argument we have that $\Omega^{s_1 \times \cdots \times s_n \to s}$ is isomorphic to

$$\{(\omega, ((\emptyset, \ldots, \emptyset), \emptyset)) \in \Omega \times (I^n \times I) \mid \\ \tau(\omega) = (e_n \circ (s_1, \ldots, s_n), s)((\emptyset, \ldots, \emptyset), \emptyset)\}$$

and the pullback requirement

$$\tau \circ p_1 = (e_n \circ (s_1, \ldots, s_n), s) \circ p_2$$

this time simply means that $\tau(\omega) = (\mathtt{s}_1 \cdots \mathtt{s}_n, \mathtt{s})$, i. e., ω is of the form $\omega \colon \mathtt{s}_1 \times \cdots \times \mathtt{s}_n \to \mathtt{s}$.

Thus, the pullback view produces signatures that correspond with the strict mathematical view of many-sorted signatures over `Set`.

Example 2.6.2. Clearly, it is now interesting to consider the concrete case of signatures over $\mathtt{Set}(\mathfrak{L})$. Let us inspect the pullback diagram

$$\begin{array}{ccc} (\Omega^{\to \mathtt{s}}, \beta) & \xrightarrow{p_1} & (\Omega, \alpha) \\ \downarrow{\scriptstyle p_2} & & \downarrow{\scriptstyle \tau} \\ (\{(\emptyset,\emptyset)\}, \top) & \xrightarrow[e_0 \otimes \mathtt{s}]{} & (\hat{S} \times S, \gamma) \end{array}$$

which, by the definition of pullback in $\mathtt{Set}(\mathfrak{L})$, gives

$$\Omega^{\to \mathtt{s}} \cong \{(\omega, (\emptyset,\emptyset)) \in \Omega \times \{(\emptyset,\emptyset)\} \mid \tau(\omega) = (e_0, \mathtt{s})(\emptyset,\emptyset)\}$$

and

$$\beta(\omega, (\emptyset,\emptyset)) = \alpha(\omega) \wedge \top(\omega) = \alpha(\omega).$$

Using the pullback condition for some operator symbol ω, gives the expected

$$\begin{aligned} \tau(\omega) &= (e_0 \otimes \mathtt{s})(\emptyset,\emptyset) = (e, \mathtt{s}) \text{ and} \\ \gamma(e, \mathtt{s}) &= \top. \end{aligned}$$

We forgo the construction for non-constants since it is, similar to the situation in `Set`, a straightforward extension of the case for constants.

With $\Sigma_1 = (\Omega_1, \tau_1, S_1)$ and $\Sigma_2 = (\Omega_2, \tau_2, S_2)$ being $\mathtt{C}$-signatures, we can now define a signature morphism $\sigma \colon \Sigma_1 \longrightarrow \Sigma_2$ as a pair of morphisms $(\mathfrak{s} \colon S_1 \longrightarrow S_2, \mathfrak{o} \colon \Omega_1 \longrightarrow \Omega_2)$ such that

$$\begin{array}{ccc} \Omega_1 & \xrightarrow{\mathfrak{o}} & \Omega_2 \\ \downarrow{\scriptstyle \tau_1} & & \downarrow{\scriptstyle \tau_2} \\ \hat{S}_1 \otimes S_1 & \xrightarrow[\hat{\mathfrak{s}} \otimes \mathfrak{s}]{} & \hat{S}_2 \otimes S_2 \end{array}$$

commutes. Here $\hat{\mathfrak{s}}$ is the monoidal extension of $\mathfrak{s}$. Thus we arrive at a category $\mathtt{Sign}_{\mathtt{C}}$ of many-sorted signatures over $\mathtt{C}$. The category $\mathtt{Sign}_{\mathtt{Set}}$ corresponds intuitively with the informally defined category of many-sorted signatures.

In the following we provide some typical examples of signatures in $\mathtt{Sign}_{\mathtt{Set}}$. We use now the computer science notation rather than the mathematical notation.

Example 2.6.3. A signature for natural numbers could be given by $\mathtt{NAT} = (\{\mathtt{nat}\}, \{\mathtt{0}\colon \to \mathtt{nat}, \mathtt{succ}\colon \mathtt{nat} \to \mathtt{nat}\})$, which indeed is a fundamental example that syntactically produce the natural numbers. This signature can be extended, e. g., to NATADD by adding to the set of operators in NAT the operator

$$+\colon \mathtt{nat} \times \mathtt{nat} \to \mathtt{nat}$$

that intuitively is reserved for addition but is still semantically or equationally undefined. In a similar fashion, logical operators could be included, and so on.

Example 2.6.4. The most basic signature for Booleans could be

$$\begin{aligned}\mathtt{BOOL} = (&\{\mathtt{bool}\},\\ &\{\mathtt{false}\colon \to \mathtt{bool}, \mathtt{true}\colon \to \mathtt{bool}\}),\end{aligned}$$

and additional operators could be introduced for Boolean operators.

Example 2.6.5. A signature

$$\begin{aligned}\mathtt{NATORD} = (&\{\mathtt{nat}, \mathtt{bool}\},\\ &\{\mathtt{0}\colon \to \mathtt{nat}, \mathtt{succ}\colon \mathtt{nat} \to \mathtt{nat},\\ &\quad \mathtt{false}\colon \to \mathtt{bool}, \mathtt{true}\colon \to \mathtt{bool},\\ &\quad \leqslant\colon \mathtt{nat} \times \mathtt{nat} \to \mathtt{bool}\}),\end{aligned}$$

could be introduced as a many-sorted situation where the operator $\leqslant$ needs both nat and bool. Again, this signature can be extended, e. g., to NATADDORD by adding the

$$+\colon \mathtt{nat} \times \mathtt{nat} \to \mathtt{nat}$$

operator. In a similar fashion, logical operators could be included, and so on.

Example 2.6.6. The set of terms for the empty signature $\Sigma = (S, \emptyset)$, $S \neq \emptyset$, consists only of variables.

2.7 TERM MONADS

In this section we continue to be two-fold with respect to computer science and mathematical notation. Given the fact that the literature in both areas is surprisingly sparse with respect to the formal construction of the inductive nature of the term functor, we start off by providing that formal construction first in the many-sorted and the crisp case. We then shift to the more general term functor construction being over underlying categories for uncertainty. One-sorted considerations of the term functor was given in Manes [102], however, without a formal inductive construction of the term functor. The more informal term set construction for one-sorted signatures was originally given in Gähler [59], and was by Eklund and Gähler [44] used to exemplify convergence structures. In Eklund et al. [48], the term construction was used as a basis for providing distributive laws in connection with the many-valued powerset monad in the one-sorted case.

2.7.1 *The General Term Functor Construction Over Monoidal Biclosed Categories*

In the following we provide the general term functor construction that works for any underlying cocomplete monoidal biclosed category $\mathtt{C}$ with tensor product $\otimes$. However, to allow for a more intuitive reading, we present the constructions specifically for the monoidal biclosed category $\mathtt{Set}$, where the tensor product is the cartesian product.

We therefore start off with having a signature $\Sigma = (S, \tau, \Omega)$ over $\mathtt{Set}$, where $\mathtt{S} = \mathrm{Hom}_{\mathtt{Set}}(I, S)$. One of the cornerstones for the construction will be the construction of a functor $\mathsf{T}_{\Sigma,\mathtt{s}} \colon \mathtt{Set}_{\mathtt{S}} \longrightarrow \mathtt{Set}$ giving

$$\mathsf{T}_{\Sigma} X_{\mathtt{S}} = (\mathsf{T}_{\Sigma,\mathtt{s}} X_{\mathtt{S}})_{\mathtt{s} \in \mathtt{S}}$$

such that the term functor $\mathsf{T}_{\Sigma} \colon \mathtt{Set}_{\mathtt{S}} \longrightarrow \mathtt{Set}_{\mathtt{S}}$ can be extended to a monad over $\mathtt{Set}_{\mathtt{S}}$.

As before, we have $\Omega^{s_1 \times \cdots \times s_n \to s}$ standing for the set of operators $\omega\colon s_1 \times \cdots \times s_n \to s$ in Ω, i.e., n-ary operators where we are explicit about the sorts. The many-sorted term functor $T_\Sigma : \mathtt{Set}_S \longrightarrow \mathtt{Set}_S$ can now be defined by induction. Firstly, for $T^1_{\Sigma,s}$, we need to introduce some further notation. For any $m \in \hat{S}$ and any $s \in S$ we define a functor $\Psi_{m,s}\colon \mathtt{Set}_S \longrightarrow \mathtt{Set}$ as follows. The special case $\Psi_{e,s}$, with e being the unit element in the monoid, is the constant object functor $((\Omega^{\to s})_{s\in S})_{\mathtt{Set}_S}$. Further, for $m = s_1 \cdots s_n$, we define

$$\Psi_{m,s}((X_t)_{t\in S}) = \Omega^{s_1 \times \cdots \times s_n \to s} \times \prod_{i=1,\ldots,n} X_{s_i},$$

for objects, and

$$\Psi_{m,s}((f_s)_{s\in S}) = \mathrm{id}_{\Omega^{s_1 \times \cdots \times s_n \to s}} \times \prod_{i=1,\ldots,n} f_{s_i}$$

for morphisms.

Note that everything above still works with the cartesian product for $\mathtt{Set}$ generalized to the tensor product for C.

We can now set

$$T^1_{\Sigma,s} = \coprod_{m\in\hat{S}} \Psi_{m,s},$$

and, for $\iota > 1$, we can then recursively proceed to define

$$T^\iota_{\Sigma,s} X_S = \coprod_{m\in\hat{S}} \Psi_{m,s}(T^{\iota-1}_{\Sigma,t} X_S \sqcup X_t)_{t\in S}),$$

where $\sqcup$ is the binary coproduct, and

$$T^\iota_{\Sigma,s} f_S = \coprod_{m\in\hat{S}} \Psi_{m,s}(T^{\iota-1}_{\Sigma,t} f_S \sqcup f_t)_{t\in S}).$$

This then allows us to define the functors T^ι_Σ by

$$T^\iota_\Sigma X_S = (T^\iota_{\Sigma,s} X_S)_{s\in S},$$

and

$$T^\iota_\Sigma f_S = (T^\iota_{\Sigma,s} f_S)_{s\in S}.$$

We can also show that there is a natural transformation

$$\Xi_\iota^{\iota+1} : \mathsf{T}_\Sigma^\iota \longrightarrow \mathsf{T}_\Sigma^{\iota+1}$$

such that $(\mathsf{T}_\Sigma^\iota)_{\iota>0}$ is an inductive system of endofunctors with $\Xi_\iota^{\iota+1}$ as its connecting maps. This natural transformation builds upon the canonical "embeddings" $j_s : X_s \longrightarrow \mathsf{T}^1_{\Sigma,s} X_S \sqcup X_s$ that define $(\xi_\iota^{\iota+1})_S$ according to

$$(\xi_1^2)_s = \coprod_{m \in \hat{S}} \Psi_{m,s}(j_s)$$

and

$$(\xi_\iota^{\iota+1})_s = \coprod_{m \in \hat{S}} \Psi_{m,s}(((\xi_{\iota-1}^\iota)_t \sqcup \mathsf{id}_{X_t})_{t \in S})$$

for $\iota > 1$, and then with $(\Xi_\iota^{\iota+1})_{X_S} = ((\xi_\iota^{\iota+1})_s)_{s \in S}$. Further, since $\mathtt{Set}_S$ is cocomplete, the inductive limit

$$\mathsf{F} = \mathrm{ind} \varinjlim \mathsf{T}_\Sigma^\iota$$

exists (in this case with ι starting from 1, not 0!), and the final term functor T_Σ is then given by

$$\mathsf{T}_\Sigma = \mathsf{F} \sqcup \mathsf{id}_{\mathtt{Set}_S}.$$

Example 2.7.1. In the case of NAT, we have

$$\mathsf{T}^1_{\mathtt{NAT}} X = \{\mathtt{0}\} \cup \{\mathtt{succ}(x) \mid x \in X\}$$

and

$$\begin{aligned}\mathsf{T}^2_{\mathtt{NAT}} X = &\{\mathtt{0}\} \cup \\ &\{\mathtt{succ}(x) \mid x \in X\} \cup \\ &\{\mathtt{succ}(\mathtt{0})\} \cup \\ &\{\mathtt{succ}(\mathtt{succ}(x)) \mid x \in X\}.\end{aligned}$$

Example 2.7.2. In the case of NATORD, with $S = \{\mathtt{nat}, \mathtt{bool}\}$, we have

$$\mathsf{T}^1_{\mathtt{NATORD},\mathtt{bool}} X_S = \{\mathtt{false}, \mathtt{true}\} \cup \{\leqslant(x_1, x_2) \mid x_1, x_2 \in X_{\mathtt{nat}}\}$$

and

$$\mathsf{T}^1_{\mathtt{NATORD},\mathtt{nat}} X_S = \{\mathtt{0}\} \cup \{\mathtt{succ}(x) \mid x \in X_{\mathtt{nat}}\},$$

and so on.

Note that, e. g., for $\leqslant(\mathtt{succ}(\mathtt{0}), \mathtt{0})$ we haven't provided any sentences or semantics giving preference to it being identified with `false` or `true`. This is done at a later stage when sentences enter the scene, and when the axioms representing properties of $\leqslant$ are introduced.

The construction above also implies that

$$\mathsf{T}_\Sigma X_S = (\mathsf{T}_{\Sigma,s} X_S)_{s \in S}$$

and that T_Σ is idempotent, or to be more precise, T_Σ is idempotent up to isomorphism, as there is a natural isomorphism between $\mathsf{T}_\Sigma \mathsf{T}_\Sigma$ and T_Σ.

The extension of the functor T_Σ to a monad over $\mathtt{Set}_S$ is enabled by $(\eta_s^{\mathsf{T}_\Sigma})_{X_s} : X_s \longrightarrow \mathsf{T}_{\Sigma,s} X_S$, such that for all $x \in X_s$,

$$(\eta_s^{\mathsf{T}_\Sigma})_{X_s}(x) = x,$$

and also $((\mu_s^{\mathsf{T}_\Sigma})_{X_s})_{s \in S} : \mathsf{T}_\Sigma \mathsf{T}_\Sigma X_S \longrightarrow (\mathsf{T}_{\Sigma,s} X_S)_{s \in S}$ as the identity, and again with 'identity' with respect to the natural isomorphism. Therefore we can say that we have natural transformations

$$\eta^{\mathsf{T}_\Sigma} : \mathrm{id}_{\mathtt{Set}_S} \longrightarrow \mathsf{T}_\Sigma \quad \text{and} \quad \mu^{\mathsf{T}_\Sigma} : \mathsf{T}_\Sigma \circ \mathsf{T}_\Sigma \longrightarrow \mathsf{T}_\Sigma$$

that give $\mathbf{T}_\Sigma = (\mathsf{T}_\Sigma, \eta^{\mathsf{T}_\Sigma}, \mu^{\mathsf{T}_\Sigma})$ as a monad.

In the general case of monoidal biclosed categories, with underlying category $\mathtt{C}$, T_Σ can also be completed to a monad over $\mathtt{C}$, because the tensor product preserves colimits, in particular inductive limits. The only open question is here whether this monad is idempotent. Also note that in the case of idempotent monads the Kleisli category is always equivalent to the Eilenberg–Moore category [125].

Monadic extensions are important for computer science applications since it enables composition of substitutions in the form of morphisms in the corresponding Kleisli category to these monads. The notation culture in computer science, as oriented more

towards using Set, expresses a term t as being within the term set $T_{\Sigma,s}X_S$, and this is frequently written briefly as $t :: s$, intuitively saying "the term t is of type s". Similarly, for a substitution $\upsilon\colon X_s \longrightarrow T_{\Sigma,s}X_S$, in programming languages we often write $x := t$, or even just $x = t$, instead of $\upsilon(x) = t$, where $x \in X_s$ and $t :: s$.

Consider now a signature morphism $\sigma = (\mathfrak{s}, \mathfrak{o})\colon \Sigma \longrightarrow \Sigma'$ with $\Sigma = (\Omega, \tau, S)$ and $\Sigma' = (\Omega', \tau', S')$. As was shown in Section 2.4 this morphism give rise to a functor $\mathfrak{S}\colon C_S \longrightarrow C_{S'}$ in the general case. Given the term functor construction it is now clearly of interest to establish a natural transformation $\widehat{\sigma}\colon T_\Sigma \longrightarrow T_{\Sigma'}\mathfrak{S}$ taking Σ terms to Σ' terms. To this end we first introduce a convenient family of mappings ψ^m for $m = s_1 \cdots s_n$ with $n \in \mathbb{N}$. For a C_S-object X_S we define $\psi^e(f_S) = \mathfrak{o}$ for the unit element e and for a general $m = s_1 \cdots s_n$, $n > 0$, we have

$$\psi^m(f_S) = \mathfrak{o} \otimes \coprod_{i=1,\ldots,n} f_{s_i}.$$

Intuitively, this mapping will, given a term, replace the operator symbol according to the signature mapping and apply f_S to each operation argument. In other words, in the case of traditional crisp terms we have

$$\begin{aligned} \psi^e(f_S)(\omega_0) &= \mathfrak{o}(\omega_0) \\ \psi^m(f_S)(\omega_n(t_1, \ldots, t_n) &= \mathfrak{o}(\omega_n)(f_{s_1}(t_1), \ldots, f_{s_n}(t_n)) \end{aligned}$$

for a constant $\omega_0 \in \Omega$ and n-ary operator $\omega_n \in \Omega$, respectively. Using ψ, we now introduce natural transformations

$$\widehat{\sigma}^\iota\colon T^\iota_\Sigma \longrightarrow T^\iota_{\Sigma'}\mathfrak{S}$$

defined by

$$\widehat{\sigma}^1_{X_S} = \coprod_{m \in \widehat{S}} \psi^m(j_S)$$

for $\iota = 1$ and

$$\widehat{\sigma}^\iota_{X_S} = \coprod_{m \in \widehat{S}} \psi^m(\widehat{\sigma}^{\iota-1}_{X_S} \sqcup j_{s_i})$$

for $\iota > 1$ and canonical injections $j_s \colon X_s \longrightarrow (\mathfrak{S}(X_S))_{\mathfrak{s}(s)}$.

We can now show that the diagram

$$\begin{array}{ccc} T_\Sigma^\iota X_S & \xrightarrow{(\Xi_\iota^{\iota+1})_{X_S}} & T_\Sigma^{\iota+1} X_S \\ \Big\downarrow{\scriptstyle \widehat{\sigma}^\iota_{X_S}} & & \Big\downarrow{\scriptstyle \widehat{\sigma}^{\iota+1}_{X_S}} \\ T_{\Sigma'}^\iota \mathfrak{S} X_S & \xrightarrow[(\Xi_\iota^{\iota+1})'_{\mathfrak{S}(X_S)}]{} & T_{\Sigma'}^{\iota+1} \mathfrak{S} X_S \end{array} \tag{4}$$

commutes for all $\iota > 0$, i. e., $\Xi'^{\iota+1}_\iota \circ \widehat{\sigma}^\iota = \widehat{\sigma}^{\iota+1} \circ \Xi_\iota^{\iota+1}$. This is done using an inductive argument, i. e., assuming that $\Xi'^{\iota}_{\iota-1} \circ \widehat{\sigma}^{\iota-1} = \widehat{\sigma}^\iota \circ \Xi^\iota_{\iota-1}$. For the case $\iota > 1$ and $m = s_1 \cdots s_n$ with $n > 0$ we then have for each $s \in S$ that

$$\begin{aligned} &(\xi'^{\iota+1}_\iota)_{\mathfrak{s}(s)} \circ \widehat{\sigma}^\iota \\ &= \coprod_{m \in \widehat{S'}} \Big(\mathrm{id}_{\Omega^{s_1 \times \cdots \times s_n \to s}} \otimes \prod_{i=1,\dots,n} (((\xi^\iota_{\iota-1})_t \sqcup \mathrm{id}_{X_t})_{t \in S}) \Big) \\ &\qquad \circ \coprod_{m \in \widehat{S}} \Big(\mathfrak{o} \otimes \prod_{i=1,\dots,n} (\widehat{\sigma}^{\iota-1}_{X_S} \sqcup j_{s_i}) \Big) \\ &= \coprod_{m \in \widehat{S}} \Big((\mathrm{id}_{\Omega^{\mathfrak{s}(s)_1 \times \cdots \times \mathfrak{s}(s)_n \to \mathfrak{s}(s)}} \circ \mathfrak{o}) \\ &\qquad \otimes \prod_{i=1,\dots,n} ((((\xi^\iota_{\iota-1})_t \sqcup \mathrm{id}_{X_t})_{t \in S}) \circ (\widehat{\sigma}^{\iota-1}_{X_S} \sqcup j_{s_i})) \Big) \\ &= \coprod_{m \in \widehat{S}} \Big((\mathfrak{o} \circ \mathrm{id}_{\Omega^{s_1 \times \cdots \times s_n \to s}}) \\ &\qquad \otimes \prod_{i=1,\dots,n} ((\widehat{\sigma}^\iota_{X_S} \sqcup j_{s_i}) \circ (((\xi^\iota_{\iota-1})_t \sqcup \mathrm{id}_{X_t})_{t \in S})) \Big) \\ &= \widehat{\sigma}^{\iota+1} \circ (\xi^{\iota+1}_\iota)_s \end{aligned}$$

While not shown here, the situations for $\iota = 1$ and $m = e$ are simpler and comparatively straightforward.

Now, since (4) holds, by Itô [81, p. 806], we have that $(\widehat{\sigma}^\iota)_{\iota>0}$ viewed as an inductive system is a morphism between the inductive systems $(T^\iota_\Sigma)_{\iota>0}$ to $(T^\iota_{\Sigma'})_{\iota>0}$. Further, this gives

$$\widehat{\sigma} = (\mathrm{ind} \varinjlim \widehat{\sigma}^\iota) \sqcup j_S$$

as the natural transformation taking terms over one signature to terms over another according to some signature morphism $\sigma\colon \Sigma \longrightarrow \Sigma'$.

Note, while this term morphism construction is general for all term functors over monoidal biclosed categories we have for traditional crisp terms over $\mathtt{Set}$ that this term morphism acts as expected. For example, if $\sigma = (\mathfrak{s}, \mathfrak{o})\colon \mathrm{NAT} \longrightarrow \Sigma'$ is some signature morphism, then a NAT-term $\mathtt{succ}(\mathtt{succ}(x))$ of sort $\mathtt{nat}$ is mapped to a term $\mathfrak{o}(\mathtt{succ})(\mathfrak{o}(\mathtt{succ})(j_{\mathtt{nat}}(x))$ of sort $\mathfrak{s}(\mathtt{nat})$. Technically we here have $j_{\mathtt{nat}}(x) = (x, \mathtt{nat})$ but if we adopt the common assumption that variable names must be distinct, i. e., all variable sets X_s are disjoint, then $j_{\mathtt{nat}}(x) = x$ is acceptable.

2.7.2 *Many-sorted Term Functors Over Underlying Categories For Uncertainty*

We now shift from $\mathtt{Set}$ as the underlying category for signatures to $\mathtt{Set}(\mathfrak{L})$, where $\mathfrak{L} = (L, \vee, \wedge)$ is a complete lattice. This construction follows, and extends to the monoidal biclosed category view, the construction given originally in Eklund et al. [53] but further developed in Eklund et al. [55, 56]. Thus, in this section we discuss how the term functor, as constructed over any monoidal biclosed category, specializes in the specific case of $\mathtt{Set}(\mathfrak{L})$.

Invoking uncertainty in signatures can now be performed, e. g., using $\mathtt{C} = \mathtt{Set}(\mathfrak{L})$. However, from computer science point of view, whereas uncertainty attached to operators can be intuitively justified, the interpretation of uncertain sorts is less clear. *Polymorphism of types*, on the other hand, means variables can appear under several type settings, in particular in situations where subtypes occur. Generalizations of many-sortedness to order-sortedness describes such subtypings, allowing the sort set to be partially ordered, and has been developed, e. g., by Goguen and Meseguer [73].

For a more conventional interpretation concerning computations and programming languages, a signature

$$\Sigma = ((\Omega, \beta), \tau, (S, \alpha))$$

in $\mathtt{Sign}_{\mathtt{Set}(\mathfrak{L})}$ would typically come with the restriction $\beta = \top$. The computer science notation for such restricted signatures over $\mathtt{Set}(\mathfrak{L})$ would be pairs $(S, (\Omega, \alpha))$, where S is a crisp set, and $\alpha\colon \Omega \longrightarrow L$ assigns uncertain values to operators. For the term monad construction we need objects

$$(\Omega^{s_1 \times \cdots \times s_n \to s}, \alpha^{s_1 \times \cdots \times s_n \to s})$$

for the operators $\omega\colon s_1 \times \cdots \times s_n \to s$ with n given, and

$$(\Omega^{\to s}, \alpha^{\to s})$$

for the constants $\omega\colon \to s$. These objects are provided by respective pullbacks using (Ω, α).

In our general term functor construction we have

$$\Psi_{m,s}((X_t)_{t\in S}) = \Omega^{s_1 \times \cdots \times s_n \to s} \otimes \bigotimes_{i=1,\ldots,n} X_{s_i},$$

and – recalling that $(A, \alpha) \otimes (B, \beta) = (A \times B, \alpha \wedge \beta)$ – this now specializes to

$$\begin{aligned}
&\Psi_{m,s}(((X_t, \delta_t))_{t\in S}) \\
&\quad = (\Omega^{s_1 \times \cdots \times s_n \to s},\ \alpha^{s_1 \times \cdots \times s_n \to s}) \otimes \bigotimes_{i=1,\ldots,n} (X_{s_i}, \delta_{s_i}) \\
&\quad = (\Omega^{s_1 \times \cdots \times s_n \to s} \times \prod_{i=1,\ldots,n} X_{s_i}, \\
&\qquad \alpha^{s_1 \times \cdots \times s_n \to s} \wedge \bigwedge_{i=1,\ldots,n} \delta_{s_i}).
\end{aligned}$$

The extension of $\mathsf{T}_\Sigma\colon \mathtt{Set}(\mathfrak{L})_S \longrightarrow \mathtt{Set}(\mathfrak{L})_S$ follows the same type of construction as compared to T_Σ over $\mathtt{Set}_S$, and we arrive at $\mathbf{T}_\Sigma = (\mathsf{T}_\Sigma, \eta^{\mathsf{T}_\Sigma}, \mu^{\mathsf{T}_\Sigma})$ as a monad over $\mathtt{Set}(\mathfrak{L})_S$.

Example 2.7.3. The uncertainty related to operators is quite intuitive, and clearly makes the distinction between operator and value of operation. This is exemplified in Eklund [43] were examples were provided for signatures representing assessment scales in old age psychiatry, where the distinction between observer and observation is important, e. g., when using assessment scales for depression.

Example 2.7.4. Uncertainty related to variables is less intuitive. Let us consider the following situation concerning the perception and use of "100 meters". A golfer A receives this information, into a variable x_A, about a distance to flag from ball position on the fairway. The green is surrounded by bunkers both behind and in front of the green, so hitting the ball 90 or 110 meters may lead to the ball landing in a bunker. This distance information together with visual identification of the shape of green is a basis for club selection. The choice may be the pitching or the sand wedge. A non-golfer B and a person not so well aware of distances in the range of 80–200 meters with a ± 5 meter accuracy receives the same information into a variable x_B, and uses that information to suggest carrying some heavy items by hand instead of waiting for a car to provide that transportation of the items. Clearly, x_A and x_B are differently typed so that "100" is also differently typed for A and B. This in effect means that the variables have a different capacity to carry and embrace that information.

2.7.3 *Substitution and Assignment*

In this subsection we again follow the computer science notations. Variables are syntactic and their values are semantic. In variable assignments we further need to distinguish between syntactic and semantic operators. Note how ω in the framework of its Σ-algebra $\mathfrak{A} = (A, \mathfrak{a})$ is hosted by Σ. However, the interpretation $\mathfrak{a}(\omega)$ of ω is also an operator but not formally hosted by a signature as compared to ω. This may lead to confusion and in to overcome this confusion we have to be more clear about the semantic operators to which syntactic operators are mapped. Now consider Σ as the syntactic hosting signature and Σ' the corresponding semantic hosting signature. Further, let $\sigma = (\mathfrak{s}, \mathfrak{o}) \colon \Sigma \longrightarrow \Sigma'$ be a signature mapping that binds syntactic operators over to operators on the algebraic side, and let $\widehat{\sigma} \colon \mathsf{T}_\Sigma \longrightarrow \mathsf{T}_{\Sigma'}$ be the corresponding natural transformation.

For X_S being a $\mathtt{Set}_S$-object of variables, $\mathfrak{A}_\Sigma = (A_S, \mathfrak{a}_S)$ being a Σ-algebra, and $\mathfrak{A}_{\mathsf{T}_\Sigma} = (A_S, h_S)$ a T_Σ-algebra, by a *variable assignment* with respect to X and $\mathfrak{A}_\Sigma$, i. e., an assignment for variables in X to values in $\mathfrak{A}_\Sigma$, we mean a morphism $(\mathfrak{v}^{X_S, \mathfrak{A}_\Sigma})_S \colon X_S \longrightarrow A_S$,

where the term $\mathfrak{A}_\Sigma$-evaluation is $T_\Sigma(\mathfrak{v}^{X_S,\mathfrak{A}_\Sigma})_S \colon T_\Sigma X_S \longrightarrow T_\Sigma A_S$. A variable assignment with respect to X_S and $\mathfrak{A}_{T_\Sigma}$, i. e., an assignment for variables in X_S to values in $\mathfrak{A}_{T_\Sigma}$, is a morphism $(\mathfrak{v}^{X_S,\mathfrak{A}_{T_\Sigma}})_S \colon X_S \longrightarrow A_S$, where the term $\mathfrak{A}_{T_\Sigma}$-evaluation is

$$h_S \circ \widehat{\sigma}_{A_S} \circ T_\Sigma(\mathfrak{v}^{X_S,\mathfrak{A}_{T_\Sigma}})_S \colon T_\Sigma X_S \longrightarrow A_S.$$

Note, $\mathfrak{v}^{X_S,\mathfrak{A}_\Sigma}$ may be written more simply as $\mathfrak{v}$ when the choice of X_S and $\mathfrak{A}_\Sigma$ is clear from context.

For a term $t_s \in T_\Sigma X_S$, note how $T_\Sigma(\mathfrak{v}^{X_S,\mathfrak{A}_{T_\Sigma}})_S(t_s)$ is the syntactic term in the appearance as its semantic counterpart, but still using its syntactical operators, and $\widehat{\sigma}_{A_S} \circ T_\Sigma \mathfrak{v}^{X,\mathfrak{A}_{T_\Sigma}}(t_s)$ is correspondingly the term in its semantic appearance using its semantic operators. Finally,

$$(h_S \circ \widehat{\sigma}_{A_S} \circ T_\Sigma(\mathfrak{v}^{X_S,\mathfrak{A}_{T_\Sigma}})_S)(t_s)$$

represents *the value of* t_s *in* A_s, as the computed value given the semantics of the operators.

2.8 COMPOSING MONADS

In this section we will return to the topic of partially ordered monads, in particular we will explore the nature of monad compositions involving partially ordered monads. We will consider composed monads as given in Beck [15] and Eklund et al. [48]. This definition, reproduced in Definition 2.8.1, state the requirements necessary when constructing a new monad based on the composition of two monads' underlying functors.

Definition 2.8.1. Given monads

$$\mathbf{F} = (F, \eta^F, \mu^F) \qquad \text{and} \qquad \mathbf{G} = (G, \eta^G, \mu^G),$$

a *distributive law* is given by a natural transformation

$$\sigma \colon G \circ F \longrightarrow F \circ G$$

such that

(i) $\sigma_{GA} \circ G\eta^{FG}_A = \eta^F_{GGA} \circ \eta^G_{GA}$

(ii) $F\mu^G_A \circ \sigma_{GA} \circ G\mu^{FG}_A = \mu^{FG}_A \circ F\mu^G_{FGA} \circ \sigma_{GFGA}$

(iii) $\sigma_A \circ \eta^G_{FA} = F\eta^G_A$.

A natural transformation satisfying these conditions is called a *swapper*.

The conditions on the swapper in a distributive law are precisely what we need so that we can compose monads to get a monad: $\mathbf{F} \bullet \mathbf{G} = (F \circ G, \eta^{FG}, \mu^{FG})$ where

$$\eta^{FG}_A = \eta^F_{GA} \circ \eta^G_A \qquad \text{and} \qquad \mu^{FG}_A = F\mu^G_A \circ \mu^F_{GGA} \circ F\sigma_{GA}.$$

Note, we often substitute $\bullet$ with simple juxtaposition, i. e., instead of $\mathbf{F} \bullet \mathbf{G}$ we often write just $\mathbf{FG}$.

Example 2.8.1. Recalling the $\mathbf{L}_S$ and $\mathbf{T}_\Sigma$ monads for a crisp many-sorted signature $\Sigma = (S, \Omega)$ we may form the composite monad $\mathbf{L}_S \bullet \mathbf{T}_\Sigma$. In the composition, the swapper σ is defined using inductive limits as $\sigma_X = \operatorname{ind}\varinjlim \sigma^\iota \sqcup \mathrm{id}_{L_S X_S}$ where $\sigma^\iota \colon T^\iota_\Sigma L_S \longrightarrow L_S T^\iota_\Sigma$ is defined for $\mathtt{Set}_S$-object X_S by

$$\sigma^1_{X_S} = \coprod_{m \in \hat{S}} \psi^m(\mathrm{id}_{L_S X_S})$$

for $\iota = 1$ and for $\iota > 1$ we have

$$\sigma^\iota_{X_S} = \coprod_{m \in \hat{S}} \psi^m(\sigma^{\iota-1}_X \sqcup \mathrm{id}_{L_S X_S}).$$

These use ψ^m defined

$$\psi^e(f_S) = \eta_{L_S X_S}$$

for the unit $m = e$ and for $m = s_1 \cdots s_n$ we have

$$\psi^m(f_S)(\omega, (t_i)_{i \leqslant n})(\omega', (t'_i)_{i \leqslant n'}) = \begin{cases} \bigwedge_{i \leqslant n} f_S(t_i)(t'_i) & \text{if } \omega = \omega' \\ \bot & \text{otherwise.} \end{cases}$$

Now, consider monad composition in which the participating monads are partially ordered. Will the composition then also be

ordered? The following proposition shows that this is indeed the case, provided the swapper uphold an ordering condition. In fact, as the proposition shows, the 'inner' monad is not required to be a partially ordered monad.

Proposition 2.8.1. *Let* $\mathbf{F} = (F, \leqslant, \eta^F, \mu^F)$ *be a partially ordered monad and* $\mathbf{G} = (G, \eta^G, \mu^G)$ *be any monad such that* $\mathbf{F} \bullet \mathbf{G}$ *exists and the swapper,* σ*, is such that* $\sigma_B \circ Gf \leqslant \sigma_B \circ Gg$ *for all* $f, g \colon A \longrightarrow FB$ *satisfying* $f \leqslant g$*. Then* $(F \circ G, \leqslant, \eta^{FG}, \mu^{FG})$ *is a partially ordered monad.*

Proof. First, to show that $(F \circ G, \leqslant, \eta^{FG})$ is a basic triple we endow $\mathbf{G}$ with some trivial partial order, e. g., the diagonal. This in turn equips $\mathbf{G}$ with an underlying basic triple. The result is then immediate from composition of basic triples.

It remains to show that morphisms $f, g \colon A \longrightarrow FGB$ such that $f \leqslant g$ imply that $\mu^{FG}_B \circ FGf \leqslant^F \mu^{FG}_B \circ FGg$ and that μ^{FG} preserves non-empty suprema. The latter is immediate since morphisms under F preserve non-empty suprema and μ^F does so by definition.

For the implication, we observe that expansion of μ^{FG} gives

$$F\mu^G_B \circ \mu^F_{GGB} \circ F\sigma_{GB} \circ FGf \leqslant^F F\mu^G_B \circ \mu^F_{GGB} \circ F\sigma_{GB} \circ FGg.$$

Using the naturality of μ^G and μ^F we rewrite to

$$\mu^F_{GB} \circ FF\mu^G_B \circ F\sigma_{GB} \circ FGf \leqslant^F \mu^F_{GB} \circ FF\mu^G_B \circ F\sigma_{GB} \circ FGg$$

that – since functors distribute over morphism composition – we may equivalently state in the form

$$\mu^F_{GB} \circ F(F\mu^G_B \circ \sigma_{GB} \circ Gf) \leqslant^F \mu^F_{GB} \circ F(F\mu^G_B \circ \sigma_{GB} \circ Gg).$$

Since $\mathbf{F}$ is a partially ordered monad it therefore suffices to show that

$$F\mu^G_B \circ \sigma_{GB} \circ Gf \leqslant^F F\mu^G_B \circ \sigma_{GB} \circ Gg.$$

The result now immediately follows by applying the proposition condition on σ and preservation of non-empty suprema of morphisms under F. □

Example 2.8.2. As was shown in Example 2.3.4, we have $\mathbf{L}$ partially ordered. We may now therefore show that the swapper σ from the composite monad $\mathbf{L} \bullet \mathsf{T}_\Omega$ upholds the condition of Proposition 2.8.1. Note, here we take T_Ω to be the many-sorted monad restricted to the one-sorted case with operators indexed directly by arity; this so as to operate directly over $\mathtt{Set}$ rather than $\mathtt{Set}_S$, which would be incompatible with the partially ordered monad construction. For brevity we here say T instead of T_Ω.

The result follows by determining that given two morphisms $f, g \colon X_S \longrightarrow L_S Y_S$ such that $f(x) \leqslant g(x)$ for all $x \in X_S$, then

$$(\sigma_{Y_S} \circ \mathsf{T}f)(t) \leqslant (\sigma_{Y_S} \circ \mathsf{T}g)(t)$$

for all $t \in \mathsf{T}X_S$.

The result is immediate if t is a variable or a constant, i. e., $t \in X_s$ or $t \in \Omega^{\to s}$ for some sort s. If, however,

$$t = (m, \omega, (t_i)_{i \leqslant n}) \quad \text{and} \quad t' = (m, \omega, (t'_i)_{i \leqslant n})$$

are terms in $\mathsf{T}^\iota X_S$ for some $\iota > 1$ and $m = s_1 \cdots s_n$ then assume

$$\sigma_{Y_S}^{\iota-1} \circ \mathsf{T}f(t_i) \leqslant \sigma_{Y_S}^{\iota-1} \circ \mathsf{T}g(t_i)$$

for each t_i. By induction this gives

$$\begin{aligned} \sigma_{Y_S}^{\iota}(\mathsf{T}f(t))(t') &= \bigwedge_{i \leqslant n} \sigma_{Y_S}^{\iota-1}(\mathsf{T}f(t_i))(t'_i) \\ &\leqslant \bigwedge_{i \leqslant n} \sigma_{Y_S}^{\iota-1}(\mathsf{T}g(t_i))(t'_i) \\ &= \sigma_{Y_S}^{\iota}(\mathsf{T}g(t))(t'). \end{aligned}$$

Thus, the composite monad $\mathbf{L} \bullet \mathsf{T}_\Omega$ is partially ordered and share its order relation with $\mathbf{L}$.

2.9 KLEENE MONADS

In this section we will produce a larger example of a novel theoretical development using partially ordered monads as a starting

point. The underlying idea is that partially ordered monads with a rather modest extension can be shown to establish Kleene algebras. This provides a generalized notion of powerset Kleene algebras extending the examples of Kleene algebras beyond the typical examples of and strings – as in the original work by Kleene [90] – and relations [133]. Put briefly, Kleene algebra is an axiomatization of the notion of regular expressions and languages. They are subsequently of great importance, e. g., in formal languages [94, 122] and analysis of algorithms [4, 92].

Let us first introduce two algebraic structures to clarify the nature of Kleene algebras. The first of these is the notion of an *idempotent semiring*, which is an algebra defined by $(K, +, \cdot, {}^*, 0, 1)$ satisfying the conditions

$$p + (q + r) = (p + q) + r \tag{5}$$
$$p + q = q + p \tag{6}$$
$$p + 0 = p \tag{7}$$
$$p + p = p \tag{8}$$
$$p \cdot (q \cdot r) = (p \cdot q) \cdot r \tag{9}$$
$$1 \cdot p = p \tag{10}$$
$$p \cdot 1 = p \tag{11}$$
$$p \cdot (q + r) = p \cdot q + p \cdot r \tag{12}$$
$$(p + q) \cdot r = p \cdot r + q \cdot r \tag{13}$$
$$0 \cdot p = 0 \tag{14}$$
$$p \cdot 0 = 0 \tag{15}$$

A *Kleene algebra*, then, is an idempotent semiring having an additional unary operator * (usually written in a post-fix position), i. e., an algebra $\mathfrak{K} = (K, +, \cdot, {}^*, 0, 1)$ satisfying the conditions

$$1 + p \cdot p^* = p^* \tag{16}$$
$$1 + p^* \cdot p = p^* \tag{17}$$
$$q + p \cdot r \leqslant r \Rightarrow p^* \cdot q \leqslant r \tag{18}$$
$$q + r \cdot p \leqslant r \Rightarrow q \cdot p^* \leqslant r \tag{19}$$

In place of (18) and (19) we may use the equivalent conditions

$$p \cdot r \leqslant r \Rightarrow p^* \cdot r \leqslant r \tag{20}$$

$$r \cdot p \leqslant r \Rightarrow r \cdot p^* \leqslant r \tag{21}$$

Definition 2.9.1. A partially ordered monad $\mathsf{F} = (\mathsf{F}, \leqslant, \eta, \mu)$ over $\mathtt{Set}$ is said to be a *Kleene monad*, if there exists a natural transformation $0\colon \mathrm{id} \longrightarrow \mathsf{F}$ such that the conditions

$$f \diamond 0_A = 0_A \tag{22}$$

$$0_A \diamond f = 0_A \tag{23}$$

are fulfilled for any morphism $f\colon \mathrm{id} \longrightarrow \mathsf{F}$. When ambiguity is a risk, we say 0^{F} rather than 0.

As will be seen later these conditions will precisely encode the requirements necessary to define a Kleene algebra over a monad. Before that, however, we demonstrate that the monad $\mathbf{L}$ fulfill these conditions.

Proposition 2.9.1. *The partially ordered monad* $\mathbf{L}$ *is a Kleene monad with* $0_X(x)\colon X \longrightarrow \mathsf{L}X$ *being the constant* 0 *function for all* X, $x \in X$.

Proof. We begin by showing that 0 is a natural transformation. We have for some function $f\colon X \longrightarrow Y$ and $y \in Y$

$$(\mathsf{L}f \circ 0_X)(y) = \bigvee_{f(x)=y} 0_X(x) = 0 = (0_X \circ f)(y).$$

We must also show that 0 uphold (22) and (23). This is done as follows.

(22) Given any function $f\colon X \longrightarrow \mathsf{L}X$ and $x, y \in X$ we have

$$\begin{aligned}(f \diamond 0_X)(x)(y) &= (\mu_X \circ \mathsf{L}0_X \circ f)(x)(y) \\ &= \bigvee_{A \in \mathsf{L}X} A(y) \wedge \left(\bigvee_{0_X(x')=A} f(x)(x') \right)\end{aligned}$$

which gives two cases. If A is the constant zero function then we have

$$\bigvee_{A \in \mathsf{L}X} 0 \wedge \left(\bigvee_{0_X(x')=A} f(x)(x') \right) = 0$$

otherwise we have

$$\bigvee_{A \in LX} A(y) \wedge 0 = 0$$

since $0_X(x') = A$ is never satisfied.

(23) Given any function $f\colon X \longrightarrow LX$ and $x, y \in X$ we have

$$\begin{aligned}(0_X \diamond f)(x)(y) &= (\mu_X \circ Lf \circ 0_X)(x)(y) \\ &= \bigvee_{A \in LX} A(y) \wedge \left(\bigvee_{f(x')=A} 0_X(x)(x') \right) \\ &= \bigvee_{A \in LX} A(y) \wedge \left(\bigvee_{f(x')=A} 0 \right) \\ &= \bigvee_{A \in LX} A(y) \wedge 0 = 0.\end{aligned}$$

All requirements from Definition 2.9.1 have been shown and we may conclude that $\mathbf{L}$ indeed is a Kleene monad. □

We have previously seen that the composition of a partially ordered monad with any other monad is itself partially ordered. This makes the composition eligible for being a Kleene monad. The following proposition establishes that this is the case, provided that is the composition swapper respects the natural transformation 0.

Proposition 2.9.2. *Let* $\mathbf{F} = (F, \leqslant^F, \eta^F, \mu^F)$ *be a Kleene monad and* $\mathbf{G} = (G, \eta^G, \mu^G)$ *be any monad such that the composition* $\mathbf{F} \bullet \mathbf{G}$ *exists. Then* $\mathbf{F} \bullet \mathbf{G}$ *is a Kleene monad with* $0 = 0^{FG} = 0^F \star \eta^G$*, provided that* $f \diamond 0^F_{GA} = 0^{FG}_A$*, for all morphisms* $f\colon A \longrightarrow FGA$*, and the swapper,* σ*, is such that* $\sigma \circ G0^F = 0^F G$*.*

Proof. Naturality of 0 is immediate from the naturality of 0^F and η^G. The remaining conditions are shown as follows.

(22) Given any function $f\colon A \longrightarrow FGA$ we have

$$\begin{aligned}
(f \diamond 0_A) &= \mu^{FG}_A \circ FG0_A \circ f \\
&= F\mu^G_A \circ \mu^F_{GGA} \circ F\sigma_{GA} \circ FG0_A \circ f \\
&= \mu^F_{GA} \circ FF\mu^G_A \circ F\sigma_{GA} \circ FG0_A \circ f \\
&= \mu^F_{GA} \circ FF\mu^G_A \circ F\sigma_{GA} \circ FGF\eta^G_A \circ FG0^F_A \circ f \\
&= \mu^F_{GA} \circ FF\mu^G_A \circ FFG\eta^G_A \circ F\sigma_A \circ FG0^F_A \circ f \\
&= \mu^F_{GA} \circ F\sigma_A \circ FG0^F_A \circ f \\
&= \mu^F_{GA} \circ F(\sigma_A \circ G0^F_A) \circ f \\
&= \mu^F_{GA} \circ F0^F_{GA} \circ f \\
&= f \diamond 0^F_{GA} \\
&= 0_A
\end{aligned}$$

(23) Given any function $f\colon A \longrightarrow FGA$ we have

$$\begin{aligned}
0_A \diamond f &= \mu^{FG}_A \circ FGf \circ 0_A \\
&= F\mu^G_A \circ \mu^F_{GGA} \circ F\sigma_{GA} \circ FGf \circ 0_A \\
&= \mu^F_{GA} \circ FF\mu^G_A \circ F\sigma_{GA} \circ FGf \circ 0_A \\
&= \mu^F_{GA} \circ F(F\mu^G_A \circ \sigma_{GA} \circ Gf) \circ 0_A \\
&= \mu^F_{GA} \circ F(F\mu^G_A \circ \sigma_{GA} \circ Gf) \circ 0^F_{GA} \circ \eta^G_A \\
&= (0^F_{GA} \diamond (F\mu^G_A \circ \sigma_{GA} \circ Gf)) \circ \eta^G_A \\
&= 0^F_{GA} \circ \eta^G_A \\
&= 0_A
\end{aligned}$$

Since all required properties are upheld, we conclude that $\mathbf{F} \bullet \mathbf{G}$ is a Kleene monad. □

We may now apply Proposition 2.9.2 in showing that the composition $\mathbf{L} \bullet \mathbf{T}_\Omega$ is a Kleene monad. Here we again take $\mathbf{T}_\Omega$ to be the many-sorted monad restricted to the one-sorted case.

Proposition 2.9.3. *The partially ordered monad* $\mathbf{L} \bullet \mathbf{T}_\Omega$ *is a Kleene monad.*

Proof. We must show that $f \diamond 0^{\mathsf{L}}_{\mathsf{T}X} = 0^{\mathsf{LT}}_{X}$, for all $f\colon X \longrightarrow \mathsf{L}\mathsf{T}X$, and that $\sigma_X \circ \mathsf{T}0^{\mathsf{L}}_{X} = 0^{\mathsf{L}}_{\mathsf{T}X}$. We begin by showing the former. Let $x \in X$, $y \in \mathsf{T}X$, and $f\colon X \longrightarrow \mathsf{L}\mathsf{T}X$ be any function, we then have

$$\begin{aligned}(f \diamond 0^{\mathsf{L}}_{\mathsf{T}X})(x)(y) &= (\mu^{\mathsf{L}}_{\mathsf{T}X} \circ \mathsf{L}0^{\mathsf{L}}_{\mathsf{T}X} \circ f)(x)(y) \\ &= \bigvee_{A \in \mathsf{L}\mathsf{T}X} A(y) \wedge \left(\bigvee_{0^{\mathsf{L}}_{\mathsf{T}X}(x') = A} f(x)(x') \right) \\ &= 0 \\ &= (0^{\mathsf{L}}_{\mathsf{T}X} \circ \eta^{\mathsf{T}}_{X})(x)(y).\end{aligned}$$

We now show that $\sigma_X \circ \mathsf{T}0^{\mathsf{L}}_{X} = 0^{\mathsf{L}}_{\mathsf{T}X}$. Let 0 denote the constant zero function. Consider now a term $t \in \mathsf{T}^0$, we have

$$(\sigma_X \circ \mathsf{T}0^{\mathsf{L}}_{X})(t) = (\mathrm{id}_{\mathsf{L}X} \circ 0^{\mathsf{L}}_{\mathsf{T}X})(t) = 0^{\mathsf{L}}_{\mathsf{T}X}(t).$$

Thus, the result holds for the base case. Let

$$t = (n, \omega, (t_i)_{t \leqslant n}), t' = (n, \omega, (t'_i)_{i \leqslant n}) \in \mathsf{T}^{\iota}X$$

with $\iota > 0$ and assume $(\sigma_X \circ \mathsf{T}0^{\mathsf{L}}_{X})(t_i) = 0^{\mathsf{L}}_{\mathsf{T}X}(t_i)$ for each t_i. By induction we have

$$\begin{aligned}\sigma_X(0^{\mathsf{L}}_{X}(t))(t') &= \bigwedge_{i \leqslant n} \sigma_X(0^{\mathsf{L}}_{X}(t_i))(t'_i) \\ &= \bigwedge_{i \leqslant n} 0^{\mathsf{L}}_{\mathsf{T}X}(t_i)(t'_i) = 0 = 0^{\mathsf{L}}_{\mathsf{T}X}(t)(t')\end{aligned}$$

which establishes our desired result. □

Having established the notion of Kleene monads we may now define a Kleene algebra over these monads. Let $1 = \eta_A$, and further, for $f_1, f_2 \in \mathrm{Hom}(A, \mathsf{F}A)$, define

$$f_1 + f_2 = f_1 \vee f_2,$$

i. e., pointwise according to $(f_1 + f_2)(x) = f_1(x) \vee f_2(x)$, and

$$f_1 \cdot f_2 = f_1 \diamond f_2$$

where $f_1 \diamond f_2 = \mu_A \circ \mathsf{F} f_2 \circ f_1$ is the composition of morphisms in the corresponding Kleisli category of $\mathbf{F}$.

A partial order $\leqslant$ on $\mathrm{Hom}(A, \mathsf{F}A)$ is defined pointwise, i. e., for $f_1, f_2 \in \mathrm{Hom}(A, \mathsf{F}A)$ we say $f_1 \leqslant f_2$ whenever $f_1(x) \leqslant f_2(x)$ for all $x \in A$. Note that $f_1 \leqslant f_2$ if and only if $f_1 + f_2 = f_2$.

Proposition 2.9.4. *Let* $\mathbf{F} = (\mathsf{F}, \leqslant, \eta, \mu)$ *be a Kleene monad. Then*

$$(\mathrm{Hom}(A, \mathsf{F}A), +, \cdot, 0, 1)$$

is an idempotent semiring.

Proof. We show each condition in turn:

(5) Follows immediately from associativity of $\vee$.

(6) Follows immediately from commutativity of $\vee$.

(7) Follows immediately from 0 being the identity element of $\vee$.

(8) Follows immediately from idempotency of suprema.

(9) Follows immediately from associativity of morphisms in the Kleisli category.

(10) By naturality of η and the definition of monads we have

$$1 \cdot f_1 = \eta_A \diamond f_1 = \mu_A \circ \mathsf{F} f_1 \circ \eta_A = \mu_A \circ \eta_{\mathsf{F}A} \circ f_1 = f_1.$$

(11) The result follows directly from the definition of monads, i. e., we have

$$f_1 \cdot 1 = f_1 \diamond \eta_A = \mu_A \circ \mathsf{F}\eta_A \circ f_1 = f_1.$$

(12) By naturality of μ together with (3) we obtain

$$\begin{aligned} f_1 \cdot (f_2 + f_3) &= \mu_A \circ \mathsf{F}(f_2 + f_3) \circ f_1 \\ &= ([\mu_A \circ \mathsf{F} f_2] + [\mu_A \circ \mathsf{F} f_3]) \circ f_1 \\ &= [\mu_A \circ \mathsf{F} f_2 \circ f_1] + [\mu_x \mathsf{F} f_3 \circ f_1] \\ &= f_1 \cdot f_2 + f_1 \cdot f_3. \end{aligned}$$

(13) By naturality of μ together with (1) we obtain

$$\begin{aligned}(f_1 + f_2) \cdot f_3 &= \mu_A \circ Ff_3 \circ (f_1 + f_2) \\ &= \mu_A \circ ([Ff_3 \circ f_1] + [Ff_3 \circ f_2]) \\ &= [\mu_A \circ Ff_3 \circ f_1] + [\mu_A \circ Ff_3 \circ f_2] \\ &= f_1 \cdot f_3 + f_2 \cdot f_3.\end{aligned}$$

(14) Follows immediately from (23) since

$$0 \cdot f_1 = 0_A \diamond f_1 = 0_A.$$

(15) Follows immediately from (22) since

$$f_1 \cdot 0 = f_1 \diamond 0_A = 0_A$$

And with each necessary condition shown to hold, we conclude that Kleene monads are idempotent semirings. □

The introduction of Kleene asterates is now obvious. For mappings $f\colon A \longrightarrow FA$, define

$$f^* = \bigvee_{k=0}^{\infty} f^k$$

where $f^0 = 1$ and $f^{k+1} = f \diamond f^k = \mu_A \circ Ff^k \circ f$. Suprema of mappings $g_i\colon A \longrightarrow Y$ is given by $(\bigvee g_i)(x) = \bigvee g_i(x)$.

Theorem 2.9.5. *Let $F = (F, \leqslant, \eta, \mu)$ be a Kleene monad. Then*

$$(\mathrm{Hom}(A, FA), +, \cdot, {}^*, 0, 1)$$

is a Kleene algebra.

Proof. The remaining conditions are proved as follows.

(16) We have

$$\begin{aligned}
1 + f \cdot f^* &= 1 \vee (\mu_A \circ Ff^* \circ f) \\
&= 1 \vee (\mu_A \circ F \bigvee_{k=0}^{\infty} f^k \circ f) \\
&\overset{(3)}{=} 1 \vee \bigvee_{k=0}^{\infty} f^k \cdot f \\
&= f^0 \vee \bigvee_{k=0}^{\infty} f^{k+1} \\
&= \bigvee_{k=0}^{\infty} f^k = f^*.
\end{aligned}$$

(17) Similarly we have

$$\begin{aligned}
1 + f^* \cdot f &= 1 \vee (\mu_A \circ Ff \circ f^*) \\
&= 1 \vee (\mu_A \circ Ff \circ \bigvee_{k=0}^{\infty} f^k) \\
&\overset{(1),(2)}{=} 1 \vee \bigvee_{k=0}^{\infty} f^k \cdot f \\
&= f^0 \vee \bigvee_{k=0}^{\infty} f^{k+1} \\
&= \bigvee_{k=0}^{\infty} f^k = f^*.
\end{aligned}$$

(20) Firstly, note that $f_1 \leqslant f_2$ implies $\mu_A \circ Ff_1 \leqslant \mu_A \circ Ff_2$, and therefore $\mu_A \circ Ff_1 \circ g \leqslant \mu_A \circ Ff_2 \circ g$, i.e., $g \cdot f_1 \leqslant g \cdot f_2$.

Therefore $f \cdot g \leqslant g$ implies $f \cdot f \cdot g \leqslant f \cdot g \leqslant g$, and then also $f^k \cdot g \leqslant g$, for all k. Thus $f \cdot g \leqslant g$ implies

$$\begin{aligned} f^* \cdot g &= \mu_A \circ Fg \circ f^* \\ &= \mu_A \circ Fg \circ \bigvee_{k=0}^{\infty} f^k \\ &= \bigvee_{k=0}^{\infty} \mu_A \circ Fg \circ f^k \\ &= \bigvee_{k=0}^{\infty} f^k \cdot g \leqslant g \end{aligned}$$

(21) Similarly, note that $f_1 \leqslant f_2$ implies $\mu_A \circ Fg \circ f_1 \leqslant \mu_A \circ Fg \circ f_2$, i. e., $f_1 \cdot g \leqslant f_2 \cdot g$, by which we will have that $g \cdot f \leqslant g$ implies $g \cdot f^k \leqslant g$, for all k. Thus $g \cdot f \leqslant g$ implies

$$\begin{aligned} g \cdot f^* &= \mu_A \circ Ff^* \circ g \\ &= \mu_A \circ F \bigvee_{k=0}^{\infty} f^k \circ g \\ &= \bigvee_{k=0}^{\infty} \mu_A \circ Ff^k \circ g \\ &= \bigvee_{k=0}^{\infty} g \cdot f^k \leqslant g \end{aligned}$$

□

2.10 KLEISLI OVER KLEISLI

We will in this final section present a simple categorical construction that offers an alternate view of working in Kleisli categories of composite monads. More specifically, we show that, given a distributive law for a pair of monads, it is possible construct a Kleisli category over a Kleisli category that exactly correspond with the Kleisli category of the composite monad.

Let $\mathbf{F} = (\mathsf{F}, \eta^{\mathsf{F}}, \mu^{\mathsf{F}})$ and $\mathbf{G} = (\mathsf{G}, \eta^{\mathsf{G}}, \mu^{\mathsf{G}})$ be composable monads over a category C, so that $\mathbf{F} \bullet \mathbf{G}$ exists with $\sigma\colon \mathsf{GF} \longrightarrow \mathsf{FG}$ being the swapper.

Define a functor $\mathsf{G}_{\mathsf{F}}\colon \mathsf{C}_{\mathbf{F}} \longrightarrow \mathsf{C}_{\mathbf{F}}$ by $\mathsf{G}_{\mathsf{F}} A = \mathsf{G}A$, and for morphisms $f\colon A \dashrightarrow B$ in $\mathsf{C}_{\mathbf{F}}$, let $\mathsf{G}_{\mathsf{F}} f = \sigma_B \circ \mathsf{G}f$.

Proposition 2.10.1. G_{F} *is a functor.*

Proof. Firstly, for some $A \in \mathsf{C}_{\mathbf{F}}$ we have

$$\mathsf{G}_{\mathsf{F}}\,\mathrm{id}_A = \mathsf{G}_{\mathsf{F}}\eta^{\mathsf{F}}_A = \sigma_A \circ \mathsf{G}\eta^{\mathsf{F}}_A = \eta^{\mathsf{F}}_{\mathsf{G}A} = \mathrm{id}_{\mathsf{G}A}$$

and G_{F} therefore preserves identities. Secondly, we show that composition distributes as per

$$\begin{aligned}
\mathsf{G}_{\mathsf{F}}(f \diamond g) &= \sigma_C \circ \mathsf{G}(f \diamond g) \\
&= \sigma_C \circ \mathsf{G}(\mu^{\mathsf{F}}_C \circ \mathsf{F}g \circ f) \\
&= \sigma_C \circ \mathsf{G}\mu^{\mathsf{F}}_C \circ \mathsf{GF}g \circ \mathsf{G}f \\
&= \mu^{\mathsf{G}}_{\mathsf{G}C} \circ \mathsf{F}\sigma_C \circ \sigma_{\mathsf{G}C} \circ \mathsf{GF}g \circ \mathsf{G}f \\
&= \mu^{\mathsf{G}}_{\mathsf{G}C} \circ \mathsf{F}\sigma_C \circ \mathsf{FG}g \circ \sigma_B \circ \mathsf{G}f \\
&= \mathsf{G}_{\mathsf{F}}(f) \diamond \mathsf{G}_{\mathsf{F}}(g).
\end{aligned}$$

Thus, G_{F} is a functor as per our proposition. □

This G_{F} can be extended to monad over $\mathsf{C}_{\mathbf{F}}$ as follows. Let $\eta^{\mathsf{G}_{\mathsf{F}}} = \eta^{\mathbf{F}\bullet\mathbf{G}}$ and $\mu^{\mathsf{G}_{\mathsf{F}}} = \eta^{\mathsf{F}}\mathsf{G} \circ \mu^{\mathsf{G}}$.

Proposition 2.10.2. $\mathbf{G}_{\mathbf{F}} = (\mathsf{G}_{\mathsf{F}}, \eta^{\mathsf{G}_{\mathsf{F}}}, \mu^{\mathsf{G}_{\mathsf{F}}})$ *is a monad over* $\mathsf{C}_{\mathbf{F}}$.

Proof. To begin with,

$$\begin{aligned}
\mathsf{G}_{\mathsf{F}}\eta^{\mathsf{G}_{\mathsf{F}}}_A \diamond \mu^{\mathsf{G}_{\mathsf{F}}}_A &= \mu^{\mathsf{F}}_{\mathsf{G}A} \circ \mathsf{F}\mu^{\mathsf{G}_{\mathsf{F}}}_A \circ \mathsf{G}_{\mathsf{F}}\eta^{\mathsf{G}_{\mathsf{F}}}_A \\
&= \mu^{\mathsf{F}}_{\mathsf{G}A} \circ \mathsf{F}\mu^{\mathsf{G}_{\mathsf{F}}}_A \circ \sigma_A \circ \mathsf{G}\eta^{\mathsf{G}_{\mathsf{F}}}_A \\
&= \mu^{\mathsf{F}}_{\mathsf{G}A} \circ \mathsf{F}\eta^{\mathsf{F}}_{\mathsf{G}A} \circ \mathsf{F}\mu^{\mathsf{G}}_A \circ \sigma_A \circ \mathsf{G}\eta^{\mathsf{F}}_{\mathsf{G}A} \circ \mathsf{G}\eta^{\mathsf{G}}_A \\
&= \mathsf{F}\mu^{\mathsf{G}}_A \circ \sigma_A \circ \mathsf{G}\eta^{\mathsf{F}}_{\mathsf{G}A} \circ \mathsf{G}\eta^{\mathsf{G}}_A \\
&= \mathsf{F}\mu^{\mathsf{G}}_A \circ \eta^{\mathsf{F}}_{\mathsf{G}A} \circ \mathsf{G}\eta^{\mathsf{G}}_A \\
&= \mathsf{F}\mu^{\mathsf{G}}_A \circ \mathsf{F}\eta^{\mathsf{G}}_{\mathsf{G}A} \circ \eta^{\mathsf{F}}_{\mathsf{G}A} \\
&= \eta^{\mathsf{F}}_{\mathsf{G}A}.
\end{aligned}$$

Further,

$$\begin{aligned}
\eta^{G_F}_{G_F A} \diamond \mu^{G_F}_A &= \mu^F_{GA} \circ F\mu^{G_F}_A \circ \eta^{G_F} \\
&= \mu^F_{GA} \circ F\eta^F_{GA} \circ F\mu^G_A \circ \eta^F_{GGA} \circ \eta^G_{GA} \\
&= F\mu^G_A \circ F\eta^G_{GA} \circ \eta^F_{GA} \\
&= \eta^F_{GA}.
\end{aligned}$$

Finally,

$$\begin{aligned}
G_F\mu^{G_F}_A \diamond \mu^{G_F}_A &= \mu^F_{GA} \circ F\mu^{G_F}_A \circ G_F\mu^{G_F}_A \\
&= \mu^F_{GA} \circ F\mu^{G_F}_A \circ \sigma_{GA} \circ G\mu^{G_F}_A \\
&= \mu^F_{GA} \circ F\eta^F_{GA} \circ F\mu^G_A \circ \sigma_{GA} \circ G\eta^F_{GA} \circ G\mu^G_A \\
&= F\mu^G_A \circ \sigma_{GA} \circ G\eta^F_{GA} \circ G\mu^G_A \\
&= F\mu^G_A \circ \eta^F_{GGA} \circ G\mu^G_A \\
&= F\mu^G_A \circ FG\mu^G_A \circ \eta^F_{GGGA} \\
&= F\mu^G_A \circ F\mu^G_{GA} \circ \eta^F_{GGGA} \\
&= F\mu^G_A \circ \eta^F_{GGA} \circ \mu^G_{GA} \\
&= F\mu^G_A \circ \mu^{G_F}_{GA} \\
&= \mu^F_{GA} \circ F\eta^F_{GA} \circ F\mu^G_A \circ \mu^{G_F}_{GA} \\
&= \mu^F_{GA} \circ F\mu^{G_F}_A \circ \mu^{G_F}_{GA} \\
&= \mu^{G_F}_{G_F A} \diamond \mu^{G_F}_A.
\end{aligned}$$

Thus showing that $\mathbf{G_F}$ is a monad. □

It is now possible to consider the Kleisli category over $\mathbf{G_F}$ and in particular its relationship with the Kleisli category over $\mathbf{F \bullet G}$. Let $[\mathsf{C}_\mathbf{F}]_\mathbf{G}$ denote the Kleisli category over $\mathbf{G_F}$. Then the objects of both $[\mathsf{C}_\mathbf{F}]_\mathbf{G}$ and $\mathsf{C}_{\mathbf{F \bullet G}}$ are simply the objects of C and since

$$\begin{aligned}
\mathrm{Hom}_{[\mathsf{C}_\mathbf{F}]_\mathbf{G}}(A, B) &= \mathrm{Hom}_{\mathsf{C}_\mathbf{F}}(A, G_F B) \\
&= \mathrm{Hom}_{\mathsf{C}_\mathbf{F}}(A, GB) \\
&= \mathrm{Hom}_{\mathsf{C}}(A, FGB) \\
&= \mathrm{Hom}_{\mathsf{C}_{\mathbf{F \bullet G}}}(A, B)
\end{aligned}$$

it is clear that the two Kleisli categories also contain the same morphisms. Further, since $\eta^{\mathsf{G}_\mathsf{F}} = \eta^{\mathsf{F}\bullet\mathbf{G}}$ by definition, the identity morphisms of the two categories are identical.

Consider now the morphisms $f\colon A \dashrightarrow B$ and $g\colon B \dashrightarrow C$ in $[\mathsf{C}_\mathsf{F}]_\mathbf{G}$, we have

$$\begin{aligned}
f \diamond g \text{ (in } [\mathsf{C}_\mathsf{F}]_\mathbf{G}) &= f \diamond \mathsf{G}_\mathsf{F} g \diamond \mu_C^{\mathsf{G}_\mathsf{F}} \text{ (in } \mathsf{C}_\mathsf{F}) \\
&= \mu^\mathsf{F}_{\mathsf{G}C} \circ \mathsf{F}\mu_C^{\mathsf{G}_\mathsf{F}} \circ (f \diamond \mathsf{G}_\mathsf{F} g) \\
&= \mu^\mathsf{F}_{\mathsf{G}C} \circ \mathsf{F}\mu_C^{\mathsf{G}_\mathsf{F}} \circ \mu^\mathsf{F}_{\mathsf{G}\mathsf{G}C} \circ \mathsf{F}\mathsf{G}_\mathsf{F} g \circ f \\
&= \mu^\mathsf{F}_{\mathsf{G}C} \circ \mathsf{F}(\eta^\mathsf{F}_{\mathsf{G}C} \circ \mu^\mathsf{G}_C) \circ \mu^\mathsf{F}_{\mathsf{G}\mathsf{G}C} \circ \mathsf{F}(\sigma_{\mathsf{G}C} \circ \mathsf{G}g) \circ f \\
&= \mu^\mathsf{F}_{\mathsf{G}C} \circ \mathsf{F}\eta^\mathsf{F}_{\mathsf{G}C} \circ \mathsf{F}\mu^\mathsf{G}_C \circ \mu^\mathsf{F}_{\mathsf{G}\mathsf{G}C} \circ \mathsf{F}\sigma_{\mathsf{G}C} \circ \mathsf{F}\mathsf{G}g \circ f \\
&= \mathsf{F}\mu^\mathsf{G}_C \circ \mu^\mathsf{F}_{\mathsf{G}\mathsf{G}C} \circ \mathsf{F}\sigma_{\mathsf{G}C} \circ \mathsf{F}\mathsf{G}g \circ f \\
&= \mu_C^{\mathsf{F}\bullet\mathbf{G}} \circ \mathsf{F}\mathsf{G}g \circ f \\
&= f \diamond g \text{ (in } \mathsf{C}_{\mathsf{F}\bullet\mathbf{G}}).
\end{aligned}$$

Thus, the two categories also have their respective composition operation in common. We therefore conclude that they are indeed the same category, that is $[\mathsf{C}_\mathsf{F}]_\mathbf{G} = \mathsf{C}_{\mathsf{F}\bullet\mathbf{G}}$.

3 GENERAL LOGICS

As a foundation for the presentation of our generalization of general logics we must provide a thorough description of the traditional notion of general logics as given by Meseguer [105]. This will be done essentially in the form of a reproduction of the definitions given in that paper, albeit with some certain embellishments and examples to help provide further intuition and understanding. We will here not cover the full wealth of general logics and instead attempt to cover a fewer number of concepts in greater depth. In particular, in this chapter we introduce *institutions*, *entailment systems*, *logics*, *proof calculi*, and *logical systems* as well as their corresponding morphisms and categories. Thus, no coverage will here be attempted of the more refined structures presented by Meseguer [105].

Finally, here we interpret the introduced structures as 'classical' in the sense that they range over crisp sets of sentences and a crisp notion of logical truth. It is, as previously noted, well known that this does not preclude the construction of some non-classical logics [3, 36, 65].

Traditionally, logics have been expressed using unsorted signatures and algebras. Further, one has often limited the logics by fixing the signature and algebra. These limitations will, following the example of Loeckx et al. [98] as well as Goguen and Burstall [71], be done away with in this chapter and both many-sorted signatures and algebras as well as signature morphisms and algebra homomorphism will be taken into account. In the chapter we will look at both semantic implication – which many will recognize from the $\models$ symbol – and syntactic implication, similarly recognizable by the $\vdash$ symbol. Their relationship with each other will be established and we will further see how to preserve their meaning while moving between logics.

3.1 INSTITUTIONS

Informally an institution establishes the relationship between signatures – or rather the strings that may be formed using a signature – and semantic implication, a relationship that often go under the name of a *model theory* [71]. The idea of representing these aspects of logic within this type of structure stems from the work of Goguen and Burstall [70] and the notation used here will to a large part follow their notation. Note that signature in this context does not necessarily mean the type of signature presented in Section 2.6, instead we will here view signature as an entirely abstract construction that may be instantiated in many different ways depending on the logic of interest.

Before we are able to construct concrete institutions we naturally need to establish what they actually are, the formal definition is given below and following it are some more informal explanations describing the more obscure aspects of the definition.

Definition 3.1.1. An *institution* $\mathcal{I} = (\mathtt{Sign}, \mathtt{Sen}, \mathtt{Mod}, \models)$ consists of

- $\mathtt{Sign}$ a category of signatures;
- $\mathtt{Sen}\colon \mathtt{Sign} \longrightarrow \mathtt{Set}$ a functor where each element of $\mathit{Sen}(\Sigma)$ is called a Σ-*sentence* or Σ-*formula*;
- $\mathtt{Mod}\colon \mathtt{Sign} \longrightarrow \mathtt{Cat}^{\mathrm{op}}$ a functor for which each category of $\mathtt{Mod}(\Sigma)$ represents the category of Σ-*models*; and
- $\models$ a family of *satisfaction relations* such that

  $$\models_\Sigma \subseteq \mathrm{Ob}(\mathtt{Mod}(\Sigma)) \times \mathtt{Sen}(\Sigma)$$

  for each $\Sigma \in \mathrm{Ob}(\mathtt{Sign})$.

Each of the $\models_\Sigma$ relations must fulfill the *satisfaction condition*

$$\mathtt{Mod}(\sigma)(M') \models_\Sigma \varphi \iff M' \models_{\Sigma'} \mathtt{Sen}(\sigma)(\varphi). \tag{24}$$

for all models $M' \in \mathrm{Ob}(\mathtt{Mod}(\Sigma))$, sentences $\varphi \in \mathtt{Sen}(\Sigma)$, and signature morphisms $\sigma\colon \Sigma \longrightarrow \Sigma'$.

Note in particular that the Mod functor translates categories of models in the opposite direction in relation to the way signatures are translated. The rationale behind this definition of Mod – which at first glance may seem peculiar – may best be illustrated by considering what would happen if the seemingly more natural functor $\mathsf{Mod}'\colon \mathtt{Sign} \longrightarrow \mathtt{Cat}$ had been chosen over $\mathsf{Mod}\colon \mathtt{Sign} \longrightarrow \mathtt{Cat}^{\mathrm{op}}$ in the definition of institution. The satisfaction condition would then have been

$$M \models_\Sigma \varphi \iff \mathsf{Mod}'(\sigma)(M) \models_{\Sigma'} \mathsf{Sen}(\sigma)(\varphi)$$

for a model $M \in \mathrm{Ob}(\mathsf{Mod}'(\Sigma))$. Now consider the situation that occurs for the signature morphisms

$$\Sigma_1 \xrightarrow{\sigma_1} \Sigma \xleftarrow{\sigma_2} \Sigma_2.$$

That is, σ_1 and σ_2 have the same codomain but different domains. It is now possible to see how a situation may arise where $M_1 \in \mathrm{Ob}(\mathsf{Mod}'(\Sigma_1))$, $\varphi_1 \in \mathsf{Sen}(\Sigma_1)$, $M_2 \in \mathrm{Ob}(\mathsf{Mod}'(\Sigma_2))$, and $\varphi_2 \in \mathsf{Sen}(\Sigma_2)$ are such that

$$\mathsf{Mod}'(\sigma_1)(M_1) = \mathsf{Mod}'(\sigma_2)(M_2) = M$$

and

$$\mathsf{Sen}(\sigma_1)(\varphi_1) = \mathsf{Sen}(\sigma_2)(\varphi_2) = \varphi.$$

But then, if $M_1 \models \varphi_1$ and $M_2 \not\models \varphi_2$, what about $M \models \varphi$? In other words, we have discovered an inconsistency. The Mod functor solves this problem by effectively reversing the signature morphisms. This may be illustrated by reusing the setup that caused the problem for Mod'. We then observe that

$$\mathsf{Mod}(\sigma_1)(M) = \mathsf{Mod}(\sigma_2)(M)$$

only if $\sigma_1 = \sigma_2$, that is, the confusing situation where two different signature morphisms will yield the same model is avoided.

In conclusion, the Mod functor as given in Definition 3.1.1 is therefore the natural choice for our notion of institution and the satisfaction condition must then be as illustrated by the figure

$$\begin{array}{ccc} \mathsf{Mod}(\sigma)(M') & \models_\Sigma & \phi \\ \downarrow & & \uparrow \\ M' & \models_\Sigma & \mathsf{Sen}(\sigma)(\phi) \end{array}$$

The Sen functor is far more straight-forward and does not pose any problems since we clearly want sentences to map in the same direction as signatures. Considering the satisfaction condition further, we see that it gives as a consequence the property that satisfiability of a sentence does not change with the underlying signature representation. In short, it does not matter what name an operation may go under, it is the *operation semantics* that is of primary importance in this case.

We often wish to view the satisfaction relation not in terms of the more abstract notion of models that we have considered up to now but rather from the point of view of the set of sentences satisfied by these models. A set Γ of this type for some model $M \in \mathsf{Mod}(\Sigma)$, $\Sigma \in \mathrm{Ob}(\mathtt{Sign})$, is determined by

$$\Gamma = \{\varphi \in \mathsf{Sen}(\Sigma) \mid M \models_\Sigma \varphi\}.$$

From this type of representation, we are able to construct a full subcategory of $\mathsf{Mod}(\Sigma)$, denoted $\mathsf{Mod}(\Sigma, \Gamma)$. This subcategory has as objects all models that satisfy the sentences in Γ. This in turn allow us to define a slightly different family of satisfaction relations.

Definition 3.1.2. Let $\mathcal{I} = (\mathtt{Sign}, \mathsf{Sen}, \mathsf{Mod}, \models)$ be an arbitrary institution and $\widetilde{\models}$ be a family of binary relations where

$$\widetilde{\models}_\Sigma \subseteq \mathsf{PSen}(\Sigma) \times \mathsf{Sen}(\Sigma)$$

for each signature $\Sigma \in \mathrm{Ob}(\mathtt{Sign})$ and where P is the regular powerset functor. Then each $\widetilde{\models}_\Sigma$ is also said to be a satisfaction relation if the condition

$$\Gamma \widetilde{\models}_\Sigma \varphi \iff M \models_\Sigma \varphi$$

holds for all $\varphi \in \mathsf{Sen}(\Sigma)$ and $M \in \mathsf{Mod}(\Sigma, \Gamma)$.

Remark. Typically we will, by abuse of notation use the symbol $\models$ also for the satisfaction relation above. I. e., instead of the special symbol $\widetilde{\models}$ we use $\models$. This poses no problems since it will always be clear from context as to the nature of the left operand.

We have in Definition 3.1.1 established what the objects in the category of institutions are, the definition of its morphisms will prove more tricky and they will therefore be omitted until the required notions have been presented. The impatient reader may find the presentation of institution morphisms in Section 3.6. The rest of this section will instead demonstrate a motivating example showing how the semantic part of concrete logics may be described in the context of institutions. For this example we choose a rather simple equational logic.

3.1.1 *Institution of Equational Logic*

As a first step towards defining this institution, it is beneficial to establish what will be considered a sentence in equational logic and how to give it an appropriate functorial representation. In other words, how should Sen in this particular logic be defined? Informally a sentence in equational logic may be thought of as being the same thing as equations in elementary algebra. That is, we are simply talking about *equality* and wish to answer questions of the type: Is *something* equal to *something else*? Of course, the answer to such a question depend intimately on the nature of the particular algebras used. As might be suspected, a sentence in the equational logic setting will also be called an *equation* whose formal definition follows.

Definition 3.1.3. A Σ-*equation* for some many-sorted signature $\Sigma = (S, \Omega)$ over Set is a triple $e = (X_S, t, t')$ where X_S is a family of Σ-variables and $t, t' \in \mathsf{T}_{\Sigma,s}(X_S)$, that is, terms of some sort $s \in S$. For simplicity, this triple may be written $(\forall X_S)\ t = t'$. If $X_S = \mathrm{var}(t) \cup \mathrm{var}(t')$, where $\mathrm{var}(t)$ denotes the family of variables occurring in t, we may simplify even further by writing $t = t'$.

Note that the signatures in equational logic are identical to the notion of 'crisp' signatures as given in Section 2.7 and subsequently the $\mathtt{Sign}$ category of the institution will be the corresponding signature category, which we here will denote $\mathtt{Sign}^{\mathrm{Eq}}$.

The inclusion of the family of variables, X_S in this case, within the definition of equation warrants a brief explanation: In short, in unsorted equational logic an equation is typically represented simply as a pair (t, t'), where t, t' are terms. It is then possible to apply the common inference (or deduction) rules that apply in equational logic – such as reflexivity, symmetry, and transitivity. It is clearly desirable to maintain the same inference rules even in the many-sorted case. Unfortunately, as shown by Goguen and Meseguer [72], simply reusing the unsorted representation of an equation will compromise the soundness of the corresponding many-sorted logic. The solution, as given in their article is to make explicit the variables of interest.

Note, no semantic meaning is placed into an equation at this point, the meaning will be provided by the satisfaction relation given in Definition 3.1.6. Fortunately, the lack of semantics does not prevent us from constructing a few simple example equations.

Example 3.1.1. Using the signature NATORD from Example 2.6.5 and X_S consisting of the $\mathtt{nat}$-variables x and y, we may, for example, construct the equations

$$(\forall X_S)\ x \leqslant y = y \leqslant x$$
$$(\forall X_S)\ \mathrm{succ}(x) \leqslant y = \mathrm{succ}(x) \leqslant \mathrm{succ}(0)$$
$$(\forall X_S)\ 0 \leqslant \mathrm{succ}(x) = \mathtt{true}.$$

It is now possible to create an appropriate $\mathtt{Sen}$ functor for our institution, taking a signature to its set of sentences, or as they are known in this case; equations. This functor will in equational logic be called $\mathtt{Sen}^{\mathrm{Eq}}$ and the following definition describes its structure.

Definition 3.1.4. Let $\mathtt{Sen}^{\mathrm{Eq}} \colon \mathtt{Sign}^{\mathrm{Eq}} \longrightarrow \mathtt{Set}$ be a functor such that

$$\mathtt{Sen}^{\mathrm{Eq}}(\Sigma) = \{(X_S, t, t') \mid t, t' \in \mathsf{T}_{\Sigma,s}(X_S), s \in S, X_S \in \mathrm{Ob}(\mathtt{Set}_S)\}$$

for any signature $\Sigma = (S, \Omega) \in \mathrm{Ob}(\mathtt{Sign}^{\mathrm{Eq}})$, and

$$\mathtt{Sen}^{\mathrm{Eq}}(\sigma)(X_S, t, t') = (\mathfrak{S}(X_S), \hat{\sigma}(t), \hat{\sigma}(t'))$$

for signature morphisms $\sigma = (\mathfrak{s}, \mathfrak{o}) \colon \Sigma \longrightarrow \Sigma'$. I. e., $\mathtt{Sen}^{\mathrm{Eq}}(\Sigma)$ is the set of all Σ-equations and $\mathtt{Sen}^{\mathrm{Eq}}(\sigma)(\varphi)$ turns a Σ-equation φ into a Σ'-equation.

By using the previously define Eilenberg–Moore category we may now introduce a functor $\mathtt{Mod}^{\mathrm{Eq}}$, taking the role of the $\mathtt{Mod}$ functor in the institution.

Definition 3.1.5. Let $\mathtt{Mod}^{\mathrm{Eq}} \colon \mathtt{Sign}^{\mathrm{Eq}} \longrightarrow \mathtt{Cat}^{\mathrm{op}}$ be a functor such that for some signature $\Sigma = (S, \Omega) \in \mathrm{Ob}(\mathtt{Sign}^{\mathrm{Eq}})$ we have

$$\mathtt{Mod}^{\mathrm{Eq}}(\Sigma) = \mathtt{Set}^{\mathbf{T}_\Sigma}.$$

and for signature morphisms $\sigma = (\mathfrak{s}, \mathfrak{o}) \colon \Sigma \longrightarrow \Sigma'$ with $\Sigma = (S, \Omega) \in \mathrm{Ob}(\mathtt{Sign}^{\mathrm{Eq}})$ we have $\mathtt{Mod}^{\mathrm{Eq}}(\sigma)$ as a functor

$$\mathtt{Mod}^{\mathrm{Eq}}(\sigma) \colon \mathtt{Set}^{\mathbf{T}_{\Sigma'}} \longrightarrow \mathtt{Set}^{\mathbf{T}_\Sigma}$$

such that

(i) if $\mathfrak{A}' = (A'_{S'}, h'_{S'})$ is a $\mathbf{T}_{\Sigma'}$-algebra, then there exists a $\mathbf{T}_\Sigma$-algebra $\mathfrak{A} = (A_S, h_S) = \mathtt{Mod}^{\mathrm{Eq}}(\sigma)(\mathfrak{A}')$ that satisfies

$$A_s = A'_{\mathfrak{s}(s)} \quad \text{and} \quad h_s = h'_{\mathfrak{s}(s)} \circ \widehat{\sigma}_{A_s}$$

for each sort $s \in S$; and

(ii) if $\mathfrak{h}' \colon \mathfrak{A}' \longrightarrow \mathfrak{B}'$ is a homomorphism between the Σ'-algebras $\mathfrak{A}'$ and $\mathfrak{B}'$, then the homomorphism $\mathfrak{h} = \mathtt{Mod}^{\mathrm{Eq}}(\sigma)(\mathfrak{h}')$ is such that

$$\mathfrak{h} \colon \mathtt{Mod}^{\mathrm{Eq}}(\sigma)(\mathfrak{A}') \longrightarrow \mathtt{Mod}^{\mathrm{Eq}}(\sigma)(\mathfrak{B}')$$

and $\mathfrak{h}_s = \mathfrak{h}'_{\mathfrak{s}(s)}$ for all $s \in S$.

Having established precisely the structures that constitute sentences and models in equational logic, we may now attempt to formalize an appropriate satisfaction relation.

Definition 3.1.6. Let $(A_S, h_S) \in \mathrm{Ob}(\mathsf{Mod}^{\mathrm{Eq}}(\Sigma))$ be an algebra for the equational signature Σ with X_S being an associated family of variables. Then the satisfaction relation of equational logic, $\models^{\mathrm{Eq}}$, is such that

$$(A_S, h_S) \models^{\mathrm{Eq}}_{\Sigma} (\forall X_S)\ t = t' \iff (h_S \circ T_\Sigma(\mathfrak{v}_S))(t) = (h_S \circ T_\Sigma(\mathfrak{v}_S))(t')$$

for equations $(\forall X_S)\ t = t'$ and assignments $\mathfrak{v}_S : X_S \longrightarrow A_S$.

We will here forgo showing that this definition of the satisfaction relation fulfill the satisfaction condition of (24), but the full details are available in, e. g., Helgesson [77].

Proposition 3.1.1. *The satisfaction relation of equational logic, as given in Definition 3.1.6, comply with the satisfaction condition. That is,*

$$\mathsf{Mod}^{Eq}(\sigma)(\mathfrak{A}') \models^{Eq}_{\Sigma} (\forall X_S)\ t = t' \iff \mathfrak{A}' \models^{Eq}_{\Sigma'} \mathsf{Sen}^{Eq}(\sigma)((\forall X_S)\ t = t')$$

for each $\Sigma = (S, \Omega), \Sigma' \in \mathrm{Ob}(\mathtt{Sign}^{Eq})$ with X_S a family of variables for Σ as well as each $\sigma : \Sigma \longrightarrow \Sigma'$, $\mathfrak{A}' \in \mathrm{Ob}(\mathsf{Mod}^{Eq}(\Sigma'))$, and equation $(\forall X_S)\ t = t' \in \mathrm{Ob}(\mathsf{Sen}^{Eq}(\Sigma))$.

All the necessary components for the description of the institution of equational logic have now been covered. To conclude the example, the complete institution for equational logic is given in the following definition.

Definition 3.1.7. Let $\mathcal{I}^{\mathrm{Eq}}$ denote the institution describing the semantic aspect of equational logic. Then

$$\mathcal{I}^{\mathrm{Eq}} = (\mathtt{Sign}^{\mathrm{Eq}}, \mathsf{Sen}^{\mathrm{Eq}}, \mathsf{Mod}^{\mathrm{Eq}}, \models^{\mathrm{Eq}})$$

with each component being as defined throughout Section 3.1.1.

3.2 ENTAILMENT SYSTEMS

Having covered semantic implication and the semantic parts of logic in general, we shall now concentrate on the syntactic parts.

We shall do this in the context of *entailment systems* and these will at this point be completely abstract. This abstraction is primarily due to the process of actually proving sentences being completely separated from the notion of entailment. In other words, an entailment system merely describes the *properties* we expect syntactic implication to have. Section 3.8 will reveal a concrete way to axiomatize the proof process by adding the notion of *proof calculi*.

Definition 3.2.1. An entailment system, $\mathcal{E}$, is constructed using a triple $\mathcal{E} = (\mathtt{Sign}, \mathtt{Sen}, \vdash)$ where

- $\mathtt{Sign}$ is a category of signatures,
- $\mathtt{Sen}$ is a functor $\mathtt{Sen} \colon \mathtt{Sign} \longrightarrow \mathtt{Set}$, and
- $\vdash$ is a family of binary relations consisting of

  $$\vdash_\Sigma \subseteq \mathsf{P}\mathtt{Sen}(\Sigma) \times \mathtt{Sen}(\Sigma)$$

  for each signature $\Sigma \in \mathrm{Ob}(\mathtt{Sign})$ where $\vdash_\Sigma$ is called a Σ-*entailment*

these are subject to the condition that each $\vdash_\Sigma$

(i) is reflexive, that is, $\{\varphi\} \vdash_\Sigma \varphi$ for each $\varphi \in \mathtt{Sen}(\Sigma)$;

(ii) is monotone, that is, $\Gamma' \vdash_\Sigma \varphi$, for all $\Gamma' \subseteq \Gamma$ such that $\Gamma \vdash_\Sigma \varphi$;

(iii) is transitive, that is, given sentences φ_i, $i \in I$, such that $\Gamma \vdash_\Sigma \varphi_i$ and, additionally, $\Gamma \cup \{\varphi_i \mid i \in I\} \vdash_\Sigma \psi$, then $\Gamma \vdash_\Sigma \psi$; and

(iv) is an $\vdash$-*translation*, meaning that, if $\Gamma \vdash_\Sigma \varphi$ then for all $\sigma \in \mathrm{Hom}_{\mathtt{Sign}}(\Sigma, \Sigma')$, it is the case that $\mathsf{P}\sigma(\Gamma) \vdash_{\Sigma'} \sigma(\varphi)$.

If $\Gamma \vdash_\Sigma \varphi$, then we say that Γ is the set of *axioms* and φ *is derivable from* Γ or, alternatively, that φ *is a logical consequence of* Γ. Note that the subscript of $\vdash_\Sigma$ may be omitted if the signature is obvious from context. Finally, if an entailment system is such that for any $\Gamma \vdash_\Sigma \varphi$, there is a finite $\Gamma_0 \subseteq \Gamma$ such that $\Gamma_0 \vdash_\Sigma \varphi$, then the entailment system is called *compact*[1].

1 The notion of compact entailment systems is similar to the notion of compact topological spaces in which each open cover has a finite subcover. See, e. g., the work of Kelley [86] and Steen and Seebach Jr. [130].

More informally, the reflexivity condition on $\vdash$ means that if we assume the validity of a sentence then we may also prove it. The monotonicity condition assure us that adding axioms only expand, or possibly leave unchanged, the set of provable sentences. The transitivity condition states that adding a derivable sentence to the set of axioms does not change which other sentences we may prove. Finally, $\vdash$-translation states that entailment remains invariant under change of signature. In other words, the conditions are such that the family of entailment relations is required to behave just as we intuitively expect it to!

Note that the structure of $\vdash$ precisely matches that of $\models$ as it was presented in Definition 3.1.2. Indeed, we may directly construct an entailment system from an institution as per the following example.

Example 3.2.1. Let $\mathcal{I} = (\mathsf{Sign}, \mathsf{Sen}, \mathsf{Mod}, \models)$ denote some institution. Using the $\models$ relation from Definition 3.1.2, we may construct an entailment system $\mathcal{I}^+ = (\mathsf{Sign}, \mathsf{Sen}, \models)$. It is then trivially true that $\models$ fulfill the list of criteria posed in Definition 3.2.1 and that $\mathcal{I}^+$ is an entailment system.

This result, albeit rather elegant, is not very useful in practice. We understand the reason behind this by considering an important motivation as to why the syntactic approach to satisfaction is of interest: It allows the possibility of mechanizing the proof process. In particular it sometimes allows the *efficient* mechanization of the proof process. Certainly a highly desirable property. However, for an entailment system constructed directly from logic semantics, as in Example 3.2.1, we have in general little hope of easily finding such an efficient mechanization, or indeed any at all. This is simply due to the possible – often unbounded – enormity of the set of axioms. For this reason, the idea of compact entailment systems is quite desirable as we are then able to at least restrict the set of axioms to be finite, which simplifies the proof process somewhat.

We will occasionally consider all sentences derivable from a set of sentences, that is, their closure under entailment. This will be done frequently enough to warrant the special notation introduced in the following definition.

Definition 3.2.2. Let Γ be a set of Σ-sentences. We may from it construct the set of *theorems* provable from Γ, denoted $\Gamma^\bullet$, by letting

$$\Gamma^\bullet = \{\varphi \mid \Gamma \vdash \varphi\}.$$

Due to the reflexivity condition, we therefore have the relationship $\Gamma \subseteq \Gamma^\bullet \subseteq \mathsf{Sen}(\Sigma)$ that may be illustrated in the context of a Venn diagram such as the one shown in Figure 1.

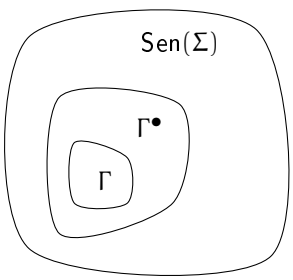


Figure 1: The relationship between Γ, $\Gamma^\bullet$, and $\mathsf{Sen}(\Sigma)$.

3.3 LOGICS

Having defined both semantic and syntactic implication – using institutions and entailment systems, respectively – we are now able to combine these two into the description of a formal *logic*.

Definition 3.3.1. A logic $\mathcal{L} = (\mathsf{Sign}, \mathsf{Sen}, \mathsf{Mod}, \vdash, \models)$ is a structure such that

(i) $(\mathsf{Sign}, \mathsf{Sen}, \vdash)$ is an entailment system;

(ii) $(\mathsf{Sign}, \mathsf{Sen}, \mathsf{Mod}, \models)$ is an institution; and

(iii) the condition

$$\Gamma \vdash_\Sigma \varphi \implies \Gamma \models_\Sigma \varphi$$

holds for all $\Sigma \in \mathrm{Ob}(\mathsf{Sign})$, $\Gamma \in \mathsf{PSen}(\Sigma)$, and $\varphi \in \mathsf{Sen}(\Sigma)$.

The last condition is commonly called the *soundness condition*. If the logic also fulfill the stronger condition

$$\Gamma \vdash_\Sigma \varphi \iff \Gamma \models_\Sigma \varphi$$

then it is said to be *complete*.

Remark. The notion of soundness could equivalently be stated by requiring that $\vdash_\Sigma \subseteq \models_\Sigma$ and similarly, if the logic is complete then $\vdash_\Sigma = \models_\Sigma$ holds for each signature $\Sigma \in \mathrm{Ob}(\mathtt{Sign})$.

Informally, the soundness condition states that it is only possible to derive things that are semantically true, which, of course is an essential property for a logic to have. The completeness condition adds the property that all semantically true sentences will also be true under syntactic implication. This is a desirable property but many times not entirely essential.

3.4 THEORIES

There are a few different, but closely related, ways of thinking about theories in this context of logic. Common to them all is that we define a theory to be a set of sentences together with the signature from which the sentences were constructed. The difference in how theories are defined stem from the way this construction is defined. The following definition – which is essentially identical to the one given in Meseguer [105] – will be used exclusively in this text.

Definition 3.4.1. A *theory* for some entailment system or institution with $\mathtt{Sign}$ being its category of signatures and $\mathtt{Sen}$, its sentence functor, is a pair $\mathsf{T} = (\Sigma, \Gamma)$ such that

- $\Sigma \in \mathrm{Ob}(\mathtt{Sign})$ is a signature, and
- $\Gamma \subseteq \mathtt{Sen}(\Sigma)$ is a set of *axioms*.

Remark. Recall that the notion of axioms also was introduced in the definition of entailment systems. As will become apparent, the two notions are very closely related and no confusion will arise from them both being referred to as "axioms".

Two other definitions of theory deserve mentioning, they both let a theory be a signature combined with a set of sentences just as in Definition 3.4.1 but instead of limiting this set of sentences to just the axioms, one states that it contains sentences closed under entailment while the other states that the set of sentences consists of those that are closed under satisfaction. This warrants the claim by Meseguer that the notion of theory presented in Definition 3.4.1 could readily be called the *presentation* of these two more expansive theory concepts [105]. In defense of using Definition 3.4.1 as the basis for theories, he states:

> However, the view of theories as presentations allows us to make finer distinctions that are important for both proof-theoretic and computational purposes. We can, for example, distinguish between a sentence that is a basic axiom and another that is a derived theorem.

As will become evident, these finer distinctions prove themselves to be quite useful indeed.

To make this abstract notion of theory more concrete, we may consider an example based on signatures defined in Section 2.6.

Example 3.4.1. Let NATORD be the signature given in Example 2.6.5. A simple and intuitive equational theory

$$\mathrm{T}_{\mathrm{NatOrd}} = (\mathrm{NATORD}, \Gamma_{\mathrm{NatOrd}})$$

might then be such that

$$\begin{aligned} \Gamma_{\mathrm{NatOrd}} = \{ & \mathtt{0} \leqslant \mathtt{succ}(x) = \mathtt{true}, && (25)\\ & \mathtt{succ}(x) \leqslant \mathtt{0} = \mathtt{false}, && (26)\\ & \mathtt{succ}(y) \leqslant \mathtt{succ}(x) = y \leqslant x\}. && (27) \end{aligned}$$

Looking closer at Definition 3.4.1, we see that with the addition of morphisms, it is possible to construct a category of theories. This category will now be defined.

Definition 3.4.2. Let $\mathcal{E}$ denote an entailment system with signature category `Sign` and sentence functor `Sen`. Then the corre-

sponding category of $\mathcal{E}$-theories, $\mathsf{Th}(\mathcal{E})$, is such that $\mathrm{Ob}(\mathsf{Th}(\mathcal{E}))$ is the set of all $\mathcal{E}$-theories and each *theory morphism*

$$\sigma_{\mathsf{Th}} : (\Sigma_1, \Gamma_1) \longrightarrow (\Sigma_2, \Gamma_2)$$

with $\Sigma_1, \Sigma_2 \in \mathrm{Ob}(\mathsf{Sign})$, $\Gamma_1 \subseteq \mathsf{Sen}(\Sigma_1)$, $\Gamma_2 \subseteq \mathsf{Sen}(\Sigma_2)$, consists of a signature morphism $\sigma : \Sigma_1 \longrightarrow \Sigma_2$ such that

$$\Gamma_2 \vdash_{\Sigma_2} \mathsf{Sen}(\sigma)(\varphi)$$

holds for each $\varphi \in \Gamma_1$. If σ_{Th} also fulfills the condition that $\mathsf{PSen}(\sigma)(\Gamma_1) \subseteq \Gamma_2$, that is, axioms are mapped to axioms, then it is said to be *axiom-preserving*. We let $\widehat{\mathsf{Th}}(\mathcal{E})$ denote the subcategory of $\mathsf{Th}(\mathcal{E})$ containing the same objects but only axiom-preserving morphisms.

Figure 2 illustrates the difference between an axiom-preserving theory morphism, σ_{Th} and a non axiom-preserving ditto, σ'_{Th}. It may be argued that the condition of axiom-preservation is quite strong, perhaps too strong, but the very nice properties it has will prove to be more valuable to us than its limitations. Besides, it is always possible to reconstruct the complete set of derivable sentences by computing the closure of the axioms under entailment.

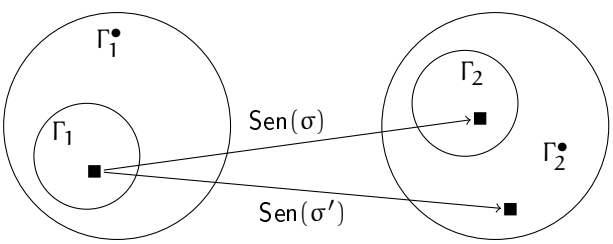


Figure 2: Comparison of axiom-preserving and non axiom-preserving theory morphisms, denoted σ_{Th} and σ'_{Th}, respectively

Since a theory embeds its associated signature and the set of axioms, it is straightforward to construct forgetful functors that given a theory extracts one or the other of these components. More specifically, we let

$$\mathcal{S}^{\mathcal{E}} : \mathsf{Th}(\mathcal{E}) \longrightarrow \mathsf{Sign} \quad \text{and} \quad \mathcal{A}^{\mathcal{E}} : \mathsf{Th}(\mathcal{E}) \longrightarrow \mathsf{Set}$$

be the forgetful functors taking a theory to its underlying signature and set of axioms, respectively. That is, for some theory, $\mathsf{T} = (\Sigma, \Gamma) \in \mathrm{Ob}(\mathtt{Th}(\mathcal{E}))$, we have $\mathcal{S}^{\mathcal{E}}(\mathsf{T}) = \Sigma$ and $\mathcal{A}^{\mathcal{E}}(\mathsf{T}) = \Gamma$. Similarly, we may construct the functor that given a signature yields the theory containing that signature but holds no axioms. I. e., let $\mathcal{T}^{\mathcal{E}} \colon \mathtt{Sign} \longrightarrow \mathtt{Th}(\mathcal{E})$ be the functor that for some signature Σ is such that $\mathcal{T}^{\mathcal{E}}(\Sigma) = (\Sigma, \emptyset)$. Note, although these functors technically are associated with a specific category of theories, and by extension some specific entailment system or institution, we will here abuse the notation and use, e. g., $\mathcal{S}$ instead of $\mathcal{S}^{\mathcal{E}}$ for any such underlying category.

The notion of theories that have been presented throughout this section will prove themselves useful when we now move towards a description on how entailment systems and institutions may be extended to categories. We begin with the category of entailment systems as its description is somewhat less complicated. Not only is it less complicated, many of the concepts needed when describing the category of entailment systems may be directly adopted in the description of the category of institutions.

3.5 THE CATEGORY OF ENTAILMENT SYSTEMS

In Section 3.2, the notion of entailment systems was presented, at that point it was not yet possible to define precisely how one moves from one entailment system to another. This will now be rectified since the necessary foundations have been presented in the last few sections. Once the mapping of one entailment system to another has been covered – and we have established that these mappings are composable – it is finally possible to formally state the definition of the category of entailment systems.

Recall that, somewhat informally, an entailment system consists of a description of the syntax of a logic – in the form of signatures and sentences over them – together with an entailment relation subject to certain conditions. Considering this, the overall idea behind an map of entailment systems is then quite intuitive. The signatures and sentence of the source logic has to be mapped to valid counterparts in the target logic. This has to be done in such a way that a derivable sentence in the source logic

becomes a derivable sentence in the target logic. Naturally a sentence that is not derivable in the source logic should be mapped to a similarly underivable sentence of the target logic.

Beginning with the mapping of signatures, we may directly observe that it is, of course, perfectly possible for the source and target entailment systems to use fundamentally different notions of signatures. By this is meant that the structure of their respective `Sign` categories may differ. It is therefore not simply a matter of applying our trusty old notion of signature morphisms, instead we need an appropriately defined functor mapping the `Sign` categories. For reasons that will become clear, a simple map of signature categories will not suffice. Instead, the functor will act on theories. Once the structure of this functor has been established, the mapping of sentences must follow. That is, the logics' respective `Sen` functor must be mapped in such a way that a valid sentence in the source logic is taken to a valid sentence in the target logic. Since this is a mapping of functors, a natural transformation is needed. The described functor, which shall be denoted Φ, together with the natural transformation, denoted α, will be sufficient for mapping entailment systems. As per the description above, Φ and α must be such that the signatures and sentences are translated correctly with respect to the relevant entailment relations.

However, before then, the formal definition of maps of entailment systems will be established. Following directly thereafter is a comprehensive informal explanation of the notions involved.

Definition 3.5.1. Let

$$\mathcal{E}_1 = (\mathsf{Sign}_1, \mathsf{Sen}_1, \vdash^1) \quad \text{and} \quad \mathcal{E}_2 = (\mathsf{Sign}_2, \mathsf{Sen}_2, \vdash^2)$$

be two entailment systems. Then a map from $\mathcal{E}_1$ to $\mathcal{E}_2$ is a pair $(\Phi, \alpha)\colon \mathcal{E}_1 \longrightarrow \mathcal{E}_2$ where

$$\Phi\colon \widehat{\mathsf{Th}}(\mathcal{E}_1) \longrightarrow \widehat{\mathsf{Th}}(\mathcal{E}_2)$$

is a functor from the category of $\mathcal{E}_1$-theories to the category of $\mathcal{E}_2$-theories, and

$$\alpha\colon \mathsf{Sen}_1 \longrightarrow \mathsf{Sen}_2 \circ \mathcal{S} \circ \Phi \circ \mathcal{T}$$

is a natural transformation such that

(i) Φ maps theory signatures with no regard to axioms, that is, $\mathcal{S}\Phi = \mathcal{S}\Phi\mathcal{T}\mathcal{S}$;

(ii) the theories $\Phi(\Sigma, \Gamma)$ and $\Phi(\Sigma, \emptyset)$ are such that

$$(\mathcal{A}\Phi(\Sigma, \Gamma))^\bullet = (\mathcal{A}\Phi(\Sigma, \emptyset) \cup \mathsf{P}\alpha_\Sigma(\Gamma))^\bullet; \text{ and}$$

(iii) for each $\Sigma \in \mathrm{Ob}(\mathtt{Sign})$,

$$\Gamma \vdash^1_\Sigma \varphi \implies \mathsf{P}\alpha_\Sigma(\Gamma) \cup \mathcal{A}\Phi(\Sigma, \emptyset) \vdash^2_{\mathcal{S}'\Phi(\Sigma,\emptyset)} \alpha_\Sigma(\varphi). \quad (28)$$

We say that the map (Φ, α) is *conservative* if instead of 3.5.1(iii), the stronger condition

$$\Gamma \vdash^1_\Sigma \varphi \iff \mathsf{P}\alpha_\Sigma(\Gamma) \cup \mathcal{A}\Phi(\Sigma, \emptyset) \vdash^2_{\mathcal{S}\Phi(\Sigma,\emptyset)} \alpha_\Sigma(\varphi)$$

holds.

Going through the conditions one by one we observe that Condition 3.5.1(i) requires the functor Φ to be such that theories with the same signature will be mapped to theories with the same signature. In other words, the output signature is uniquely defined by the input signature regardless of the particular axioms involved.

Condition 3.5.1(ii) states that the closure under entailment of the axioms translated by α together with the axioms introduced by Φ matches the corresponding closure under entailment of the axioms of the target logic. Put simply, α and Φ must be in agreement about which logic they are mapping to. A functor upholding conditions 3.5.1(i) and 3.5.1(ii) is said to be *α-sensible*.

Finally, condition 3.5.1(iii) states that Φ and α must be such that they together map signatures and sentences in a manner consistent with the logics' respective entailment relations. That is, the provability of the mapped sentence in the target logic must exactly reflect the provability of the corresponding sentence in the source logic. This condition could therefore be equivalently stated by requiring that

$$\varphi \in \Gamma^\bullet \implies \alpha_\Sigma(\varphi) \in (\mathcal{A}\Phi(\Sigma, \emptyset) \cup \mathsf{P}\alpha_\Sigma(\Gamma))^\bullet \quad (29)$$

holds and in practice this condition is often easier to work with – however, it is perhaps not as intuitive as the one given in (28).

Let us make the definition more concrete by considering a simple, but interesting, example of a map of entailment systems. This example constructs the map from an entailment system to its closure under entailment.

Example 3.5.1. Let $\mathcal{E} = (\mathtt{Sign}, \mathtt{Sen}, \vdash)$ be an entailment system, we may then form a map of entailment systems $(\mathtt{E}, \mathrm{id}_{\mathtt{Sen}}) \colon \mathcal{E} \longrightarrow \mathcal{E}$ where $\mathtt{E} \colon \widehat{\mathtt{Th}}(\mathcal{E}) \longrightarrow \widehat{\mathtt{Th}}(\mathcal{E})$ is a functor such that for some $\mathcal{E}$-theory $\mathtt{T} = (\Sigma, \Gamma)$, $\mathtt{E}(\mathtt{T})$ is the theory $\mathtt{E}(\mathtt{T}) = (\Sigma, \Gamma^\bullet)$ and $\mathrm{id}_{\mathtt{Sen}}$ is the obvious identity natural transformation.

Illustrating this map in the context of (28) gives for each $\Sigma \in \mathrm{Ob}(\mathtt{Sign})$

$$\Gamma \vdash_\Sigma \varphi \implies \Gamma^\bullet \vdash_\Sigma \varphi,$$

that is, the set of axioms is expanded to contain all provable sentences. It is easy to see that this map is actually conservative, that is, it is possible to change the logical implication to a logical equivalence.

From this definition we may then establish composite maps of entailment systems by pairwise composition of the maps' subcomponents. That is, let

$$(\Phi, \alpha) \colon \mathcal{E}_1 \longrightarrow \mathcal{E}_2 \quad \text{and} \quad (\Phi', \alpha') \colon \mathcal{E}_2 \longrightarrow \mathcal{E}_3$$

be maps of entailment systems, then the composite map

$$(\Phi', \alpha') \circ (\Phi, \alpha) \colon \mathcal{E}_1 \longrightarrow \mathcal{E}_3$$

is defined by

$$(\Phi', \alpha') \circ (\Phi, \alpha) = (\Phi' \circ \Phi, \alpha'_{\mathcal{S}, \Phi\mathcal{T}} \circ \alpha). \tag{30}$$

Of course, we also have to show that this composition is, itself, a map of entailment systems. In order to do this we will need a simple lemma.

Lemma 3.5.1. *Let* $(\Phi, \alpha)\colon \mathcal{E} \longrightarrow \mathcal{E}'$, *with* $\mathcal{E} = (\mathsf{Sign}, \mathsf{Sen}, \vdash)$, *be a map of entailment systems. Then, for each* $\Sigma \in \mathrm{Ob}(\mathsf{Sign})$, $\Gamma, \Delta \in \mathsf{PSen}(\Sigma)$, *and* $\varphi \in \mathsf{Sen}(\Sigma)$, *the implication*

$$\Gamma \cup \Delta \vdash_\Sigma \varphi \implies \mathsf{P}\alpha_\Sigma(\Gamma) \cup \mathcal{A}\Phi(\Sigma, \Delta) \vdash'_{\mathcal{S}\Phi(\Sigma,\Delta)} \alpha_\Sigma(\varphi) \quad (31)$$

holds. Further, if (Φ, α) *is conservative then the two sides are equivalent.*

Proof. Expressing (31) in the style of (29) gives

$$\varphi \in (\Gamma \cup \Delta)^\bullet \implies \alpha_\Sigma(\varphi) \in (\mathcal{A}\Phi(\Sigma, \Delta) \cup \mathsf{P}\alpha_\Sigma(\Gamma))^\bullet \quad (32)$$

and this is the statement we wish to prove. In essence, the result follows directly from the conditions in Equation 3.5.1. We know, since (Φ, α) is a map of entailment systems, that

$$\varphi \in (\Gamma \cup \Delta)^\bullet \implies \alpha_\Sigma(\varphi) \in (\mathcal{A}\Phi(\Sigma, \emptyset) \cup \mathsf{P}\alpha_\Sigma(\Gamma \cup \Delta))^\bullet \quad (33)$$

Simple derivation gives

$$\begin{aligned}
&(\mathcal{A}\Phi(\Sigma, \emptyset) \cup \mathsf{P}\alpha_\Sigma(\Gamma \cup \Delta))^\bullet \\
&\quad = \alpha_\Sigma(\varphi) \in (\mathcal{A}\Phi(\Sigma, \emptyset) \cup \mathsf{P}\alpha_\Sigma(\Gamma) \cup \mathsf{P}\alpha_\Sigma(\Delta))^\bullet \\
&\quad = \alpha_\Sigma(\varphi) \in (\mathcal{A}\Phi(\Sigma, \Delta) \cup \mathsf{P}\alpha_\Sigma(\Gamma))^\bullet \qquad \text{by 3.5.1(ii).}
\end{aligned}$$

That when inserted in (33) gives exactly (32), which was the sentence to prove. The proof for equivalence is analogous. □

In short, the above lemma states that we may move axioms arbitrarily between the application of α and the application of Φ with no risk of altering the entailment relation.

Proposition 3.5.2. *The composition of entailment systems is itself a map of entailment systems.*

Proof. Let $(\Phi, \alpha)\colon \mathcal{E}_1 \longrightarrow \mathcal{E}_2$ and $(\Phi', \alpha')\colon \mathcal{E}_2 \longrightarrow \mathcal{E}_3$ be maps of entailment systems and let $(\Phi^\circ, \alpha^\circ)$ denote the composition

$$(\Phi', \alpha') \circ (\Phi, \alpha)\colon \mathcal{E}_1 \longrightarrow \mathcal{E}_3,$$

that is, $\Phi^\circ = \Phi' \circ \Phi$ and $\alpha^\circ = \alpha'_{\mathcal{S}'\Phi\mathcal{T}} \circ \alpha$. To prove the lemma, we wish to show that the conditions of placed on maps of entailment systems are upheld. These are shown in turn.

- Condition 3.5.1(i) follows directly from the fact that Φ and Φ' follow 3.5.1(i). Thus, we get the derivation

$$\begin{aligned}\mathcal{S}\Phi^\circ = \mathcal{S}\Phi'\Phi = \mathcal{S}\Phi'\mathcal{T}\mathcal{S}\Phi \\ = \mathcal{S}\Phi'\mathcal{T}\mathcal{S}\Phi\mathcal{T}\mathcal{S} = \mathcal{S}\Phi'\Phi\mathcal{T}\mathcal{S} = \mathcal{S}\Phi^\circ\mathcal{T}\mathcal{S}.\end{aligned}$$

- Similarly, that 3.5.1(ii) holds for Φ° follows from the fact that it holds for Φ and Φ'. Let $\Phi(\Sigma,\emptyset) = (\Sigma',\emptyset')$. Using the identities $\mathsf{P}(f\circ g) = \mathsf{P}f\circ\mathsf{P}g$ and $\mathcal{S}\Phi\mathcal{T}\Sigma = \Sigma'$ as well as Lemma 3.5.1 we find that

$$\begin{aligned}&(\mathcal{A}\Phi^\circ(\Sigma,\emptyset)\cup\mathsf{P}\alpha^\circ_\Sigma(\Gamma))^\bullet \\ &\quad= (\mathcal{A}\Phi'\Phi(\Sigma,\emptyset)\cup\mathsf{P}(\alpha'_{\Sigma'}\circ\alpha_\Sigma)(\Gamma))^\bullet \\ &\quad= (\mathcal{A}\Phi'(\Sigma',\emptyset')\cup\mathsf{P}\alpha'_{\Sigma'}(\mathsf{P}\alpha_\Sigma(\Gamma)))^\bullet \\ &\quad= (\mathcal{A}\Phi'(\Sigma',\mathsf{P}\alpha_\Sigma(\Gamma)\cup\emptyset'))^\bullet \qquad \text{by Lemma 3.5.1} \\ &\quad= (\mathcal{A}\Phi'\Phi(\Sigma,\Gamma))^\bullet \\ &\quad= (\mathcal{A}\Phi^\circ(\Sigma,\Gamma))^\bullet.\end{aligned}$$

- Condition 3.5.1(iii), when rewritten as per (29) becomes

$$\varphi\in\Gamma^\bullet \implies \alpha^\circ_\Sigma(\varphi)\in(\mathcal{A}\Phi^\circ(\Sigma,\emptyset)\cup\mathsf{P}\alpha^\circ_\Sigma(\Gamma))^\bullet.$$

  Given $\Phi(\Sigma,\emptyset) = (\Sigma',\emptyset')$, then when expanded further and rewritten slightly using the same technique as above we get the equivalent statement

$$\varphi\in\Gamma^\bullet \implies \alpha'_{\Sigma'}(\alpha_\Sigma(\varphi))\in(\mathcal{A}\Phi'(\Sigma',\emptyset)\cup\mathsf{P}\alpha'_{\Sigma'}(\mathsf{P}\alpha_\Sigma(\Gamma)\cup\emptyset'))^\bullet.$$

  Since both (Φ,α) and (Φ',α') are maps of entailment systems it follows directly that this implication hold.

Thus, Φ° and α° are in compliance with the conditions of Equation 3.5.1 and subsequently, $(\Phi^\circ,\alpha^\circ) = (\Phi',\alpha')\circ(\Phi,\alpha)$ is a map of entailment systems. □

In addition to the above result, it is not difficult to see that the composition of two conservative maps of entailment systems is itself conservative. Neither is it hard to see that the composition of maps entailment system is associative; it follows from the

associativity of functor composition and vertical composition of natural transformations, respectively.

These results regarding the composition of entailment system maps were the final brick needed in the construction of the category of entailment systems. It is now a simple matter to collect these bricks into the following trivial definition:

Definition 3.5.2. Let Ent denote the category of entailment systems. Its objects are all entailment systems as presented in Definition 3.2.1 and its morphisms are maps of entailment systems as presented in Equation 3.5.1.

This concludes the treatment of the category of entailment systems, we will now look at the corresponding category for institutions.

3.6 THE CATEGORY OF INSTITUTIONS

Recall from Section 3.1 that an institution contains structures representing both the syntax of a logic and its model-theoretic aspects. That is, not only does a *map of institutions* have to translate sentences and signatures, it is also required to translate models. Fortunately, the mapping of a logic's syntactic aspects has already been covered in the description of maps of entailment systems and as will be shown, it is possible to directly adopt these ideas in the institution case.

Informally, the mapping of a logic's model-theoretic aspects must be such that the invariance of truth is guaranteed. Obviously, it is desirable to avoid maps that take a model with one semantics to a model with a completely different semantics! Note that this will be a map between two Mod functors, that is, a natural transformation is called for. Additionally, similar to how Mod maps models in reverse, our natural transformation will map the Mod functors in reverse. The underlying reason for this is the same as the previously discussed reason why the Mod functor maps models in the reverse direction, that is, we wish to avoid ending up in inconsistent situations. The natural transformation that will perform this task shall be denoted β.

First, the formal definition will be given then a few concrete examples will illustrate the mechanics of this map.

Definition 3.6.1. Let

$$\mathcal{I}_1 = (\mathsf{Sign}_1, \mathsf{Sen}_1, \mathsf{Mod}_1, \models^1)$$
$$\mathcal{I}_2 = (\mathsf{Sign}_2, \mathsf{Sen}_2, \mathsf{Mod}_2, \models^2)$$

be two institutions. Then a map from $\mathcal{I}_1$ to $\mathcal{I}_2$ is a triple

$$(\Phi, \alpha, \beta) \colon \mathcal{I}_1 \longrightarrow \mathcal{I}_2$$

where

$$\Phi \colon \widehat{\mathsf{Th}}(\mathcal{I}_1) \longrightarrow \widehat{\mathsf{Th}}(\mathcal{I}_2)$$

is an α-sensible functor from the category of $\mathcal{I}_1$-theories to the category of $\mathcal{I}_2$-theories, and the natural transformations

$$\alpha \colon \mathsf{Sen}_1 \longrightarrow \mathsf{Sen}_2 \circ \mathcal{S} \circ \Phi \circ \mathcal{T}$$
$$\beta \colon \mathsf{Mod}_2 \circ \mathcal{S} \circ \Phi \circ \mathcal{T} \longrightarrow \mathsf{Mod}_1$$

are such that the condition

$$M' \models^2_\Sigma \alpha_\Sigma(\varphi) \iff \beta_\Sigma(M') \models^1_\Sigma \varphi \tag{34}$$

hold for each $\Sigma \in \mathrm{Ob}(\mathsf{Sign})$, $\varphi \in \mathsf{Sen}(\Sigma)$, and $M' \in \mathsf{Mod}_2(\Sigma')$ with $\Sigma' = \mathcal{S}\Phi\mathcal{T}(\Sigma)$.

Remark. Recall that an α-sensible functor is such that the conditions 3.5.1(i) and 3.5.1(ii) are upheld. In other words, these conditions are implicitly required for the map of institutions as well.

Remark. Note that the Φ functor in the above definition actually does more work than is required. It is easy to see that all that is really needed for the map of institutions is a functor between the respective signature category of the institutions. However, by using a functor between the categories of theories instead, the extension to a map of logics will be very natural. This will become clear in Section 3.7.

As an example, let us consider a mapping from the many-sorted equational institution, $\mathcal{I}^{Eq}$, to its unsorted variant. This type of map is by its nature lossy, that is, some of the structure of the many-sorted institution will be lost. But the example is, nevertheless, interesting and illuminating. We will here only give a short informal description of the unsorted equational institution as its structure is quite simple and its use will become obvious from the example that follow. First, let

$$\mathcal{I}^{1Eq} = (\mathtt{Sign}^{1Eq}, \mathtt{Sen}^{1Eq}, \mathtt{Mod}^{1Eq}, \models^{1Eq})$$

be this institution. Then a signature $\Pi \in \mathrm{Ob}(\mathtt{Sign}^{1Eq})$ is a family of operations indexed by a natural number that indicates the operation arity, it may of course have an associated set of variables. The sentences are equations of unsorted and unquantified terms, that is, the set of variables does not have to be mentioned in the equation. The models are unsorted algebras where an algebra $A \in \mathrm{Ob}(\mathtt{Mod}^{1Eq}(\Pi))$ consists of a carrier set, A, and for each operation $\omega \in (\Pi_n)_{n\in\mathbb{N}}$, a function $A_\omega \colon A^n \longrightarrow A$, where A^n is making use of the familiar notation where

$$A^n = \underbrace{A \times A \times \cdots \times A}_{\text{n times}}.$$

Finally, using the obvious inductive definition of the evaluation function $A(\mathfrak{v})$, the satisfaction relation is such that for all unsorted signatures Π we have

$$A \models^{1Eq}_{\Pi} t = t' \iff A(\mathfrak{v})(t) = A(\mathfrak{v})(t')$$

for all equations $(t = t') \in \mathtt{Sen}^{1Eq}(\Pi)$, unsorted algebras $A \in \mathrm{Ob}(\mathtt{Mod}^{1Eq}(\Pi)$, and variable assignments $\mathfrak{v} \colon X \longrightarrow A$.

Example 3.6.1. Put shortly, the map to describe is

$$(\Phi, \alpha, \beta) \colon \mathcal{I}^{Eq} \longrightarrow \mathcal{I}^{1Eq}.$$

Let $\widehat{\mathtt{Th}}(\mathcal{I}^{Eq})$ denote the category of many-sorted equational theories and, similarly, let $\widehat{\mathtt{Th}}^{1Eq}$ denote the category of unsorted

equational theories. Then the functor $\Phi\colon \widehat{\mathsf{Th}}(\mathcal{I}^{\mathrm{Eq}}) \longrightarrow \widehat{\mathsf{Th}}(\mathcal{I}^{1\mathrm{Eq}})$ is such that

$$\Phi((S,\Omega),\Gamma) = (\Pi, \mathsf{P}\alpha_{(S,\Omega)}(\Gamma)),$$

where an operation $\omega \in \Pi_n$ if $\omega \in \Omega^{s_1\times\cdots\times s_n\to s}$ for some $s, s_1, \ldots, s_n \in S$ and $n \in \mathbb{N}$, in other words, the sorts that are operated upon are forgotten and only the operation name and rank is retained. The natural transformation

$$\alpha\colon \mathsf{Sen}^{\mathrm{Eq}} \longrightarrow \mathsf{Sen}^{1\mathrm{Eq}}\mathcal{S}\Phi\mathcal{T}$$

is such that for each many-sorted equation $(\forall X_S)\ t = t'$,

$$\alpha_{(S,\Omega)}((\forall X_S)\ t = t') = (u = u')$$

where u, u' are the same as t, t' but with operations replaced by their unsorted counterparts. Finally, the natural transformation

$$\beta\colon \mathsf{Mod}^{1\mathrm{Eq}}\mathcal{S}\Phi\mathcal{T} \longrightarrow \mathsf{Mod}^{\mathrm{Eq}}$$

is such that, for a many-sorted signature $\Sigma = (S,\Omega) \in \mathrm{Ob}(\mathtt{Sign}^{\mathrm{Eq}})$ and corresponding unsorted signature $\Pi = \mathcal{S}\Phi\mathcal{T}(\Sigma)$, there is for each Π-algebra A an associated many-sorted algebra $A' = \beta_\Sigma(A)$. This algebra has for each $s \in S$, the carrier set $A'(s) = A$. Additionally, for each function $A_\omega\colon A^n \longrightarrow A$ and operation $\omega \in \Omega^{s_1\times\cdots\times s_n\to s}$, $s, s_1, \ldots, s_n \in S$, $n \in \mathbb{N}$, the function

$$A'(\omega)\colon A'(s_1)\times\cdots\times A'(s_n) \longrightarrow A'(s)$$

is such that $A'(\omega) = A_\omega$. From this construction, extended to a T_Σ-algebra, it directly follows that

$$A \models^{1\mathrm{Eq}}_{\Pi} \alpha_\Sigma(\varphi) \iff \beta_\Sigma(A) \models^{\mathrm{Eq}}_{\Sigma} \varphi.$$

To make the example more concrete, consider the many-sorted signature

$$\begin{aligned}\Sigma = (&\{\mathtt{nat}, \mathtt{bool}\},\\ &\{\mathtt{0}\colon \to \mathtt{nat},\\ &\ \mathtt{succ}\colon \mathtt{nat} \to \mathtt{nat},\\ &\ \mathtt{iszero}\colon \mathtt{nat} \to \mathtt{bool}\})\end{aligned}$$

and the many-sorted equation $\varphi = ((\forall X_S)\ x = \mathtt{succ}(\mathtt{succ}(x)))$. Finally, let the unsorted algebra A be such that $A = \{☺, ☹\}$, $A_{\mathtt{0}} = ☹$, $A_{\mathtt{iszero}}(x) = ☺$, and

$$A_{\mathtt{succ}}(x) = \begin{cases} ☺ & \text{if } x = ☹ \\ ☹ & \text{if } x = ☺. \end{cases}$$

Then, using the description above, we have $\Phi(\Sigma, \Gamma) = (\Pi, \mathsf{P}\alpha_\Sigma(\Gamma))$ for each set of axioms $\Gamma \in \mathsf{PSen}^{\mathrm{Eq}}(\Sigma)$ and where Π is such that $\mathtt{0} \in \Pi_0$, $\mathtt{succ} \in \Pi_1$, and $\mathtt{iszero} \in \Pi_1$. Further, $\alpha_\Sigma(\varphi) = (x = \mathtt{succ}(\mathtt{succ}(x)))$ and $A' = \beta_\Sigma(A)$ is such that

$$\begin{aligned} A'(\mathtt{nat}) &= \{☺, ☹\} \\ A'(\mathtt{bool}) &= \{☺, ☹\} \\ A'(\mathtt{0}\colon \to \mathtt{nat}) &= ☹ \\ A'(\mathtt{iszero}\colon \mathtt{nat} \to \mathtt{bool})(x) &= ☺ \\ A'(\mathtt{succ}\colon \mathtt{nat} \to \mathtt{nat})(x) &= \begin{cases} ☺ & \text{if } x = ☹ \\ ☹ & \text{if } x = ☺. \end{cases} \end{aligned}$$

Is it now the case that

$$A \models^{\mathrm{IEq}}_{\Pi} x = \mathtt{succ}(\mathtt{succ}(x)) \iff A' \models^{\mathrm{Eq}}_{\Sigma} (\forall X_S)\ x = \mathtt{succ}(\mathtt{succ}(x)) \quad (35)$$

holds? Since there are only two possible assignments for the variable x, it is easy to check each possibility. If $\mathfrak{v}\colon X \longrightarrow A$, $x \in X_{\mathtt{nat}}$, and $\mathfrak{v}(x) = ☺$, then

$$\begin{aligned} A(\mathfrak{v})(x) = ☺ &= \mathtt{succ}(☹) \\ &= \mathtt{succ}(\mathtt{succ}(☺)) = A(\mathfrak{v})(\mathtt{succ}(\mathtt{succ}(x))). \end{aligned}$$

Dually, if $\mathfrak{v}(x) = ☹$, then

$$\begin{aligned} A(\mathfrak{v})(x) = ☹ &= \mathtt{succ}(☺) \\ &= \mathtt{succ}(\mathtt{succ}(☹)) = A(\mathfrak{v})(\mathtt{succ}(\mathtt{succ}(x))). \end{aligned}$$

In other words, the left-hand side of the equivalence in (35) is true in each case. By performing a similar check of the right-hand side we will find that it also is true in both possible cases. Subsequently, we conclude that (35) holds.

As an interesting alternative to the previous example, we may consider the reverse maps. That is, the map from the unsorted equational logic to its many-sorted counterpart. To illustrate the structure of this map, a brief description is provided in the following example.

Example 3.6.2. The map in question is $(\Phi, \alpha, \beta)\colon \mathcal{I}^{1Eq} \longrightarrow \mathcal{I}^{Eq}$ and it is simply such that $\Phi(\Pi, \Gamma) = (\Sigma, \mathsf{P}\alpha_\Pi(\Gamma))$ where, using a newly introduced sort s,

$$\Sigma = (\{s\}, \{\omega\colon s^n \longrightarrow s \mid \omega \in \Pi_n, n \in \mathbb{N}\}).$$

For some unsorted signature with associated set of variables X we let

$$\alpha_\Pi(t = t') = ((\forall Y_{\{s\}})\ u = u')$$

for each $(t = t') \in \mathsf{Sen}^{1Eq}(\Pi)$ and where $Y_s = X$ and u, u' are identical to t, t' but with each unsorted operation exchanged to its many-sorted namesake. Finally, given some unsorted signature Π, the natural transformation β is such that for any many-sorted Σ-algebra A with $\Sigma = \mathcal{S}\Phi\mathcal{T}(\Pi)$, $A' = \beta_\Pi(A)$ is an unsorted Π-algebra such that $A' = A(s)$ and having for each function $A(\omega)\colon A(s)^n \longrightarrow A(s)$, the function $A'_\omega\colon A'^n \longrightarrow A'$ defined by $A'_\omega = A(\omega)$. From this construction we then have

$$A \models^{Eq}_\Sigma \alpha_\Pi(\varphi) \iff \beta_\Pi(A) \models^{1Eq}_\Pi \varphi.$$

To conclude the example, we show that β is a natural transformation. That is, that the diagram

$$\begin{array}{ccc} \mathsf{Mod}^{Eq}\mathcal{S}\Phi\mathcal{T}\Pi & \xrightarrow{\beta_\Pi} & \mathsf{Mod}^{1Eq}\Pi \\ \Big\downarrow{\scriptstyle \mathsf{Mod}^{Eq}\mathcal{S}\Phi\mathcal{T}\sigma} & & \Big\downarrow{\scriptstyle \mathsf{Mod}^{1Eq}\sigma} \\ \mathsf{Mod}^{Eq}\mathcal{S}\Phi\mathcal{T}\Pi' & \xrightarrow[\beta_{\Pi'}]{} & \mathsf{Mod}^{1Eq}\Pi' \end{array}$$

commutes for all morphisms $\sigma\colon \Pi' \longrightarrow \Pi$, or stated equivalently, that

$$\mathsf{Mod}^{\mathsf{lEq}}\sigma \circ \beta_{\Pi} = \beta_{\Pi'} \circ \mathsf{Mod}^{\mathsf{Eq}}\mathcal{SF}\mathcal{T}\sigma \tag{36}$$

holds. Here $\sigma\colon \Pi' \longrightarrow \Pi$ is a map of unsorted signatures and have the obvious semantics, that is, each operation $\omega \in \Pi'_n$, $n \in \mathbb{N}$, is mapped to the operation $\sigma(\omega) \in \Pi_n$. Let A be some many-sorted algebra constructed from Π as per the earlier description. Then expansion of the left-hand side of (36) yield

$$\mathsf{Mod}^{\mathsf{lEq}}(\sigma)(\beta_{\Pi}(A)) = \mathsf{Mod}^{\mathsf{lEq}}(\sigma)(A') = A''$$

where A' is the unsorted Π-algebra as computed by β_{Π}. Using the intuitive definition of $\mathsf{Mod}^{\mathsf{lEq}}(\sigma)$ we then have $A(s) = A' = A''$ and $A(\omega) = A'_{\omega} = A''_{\sigma(\omega)}$ for each operation $\omega \in \Pi'$. Turning our attention to the right-hand side of (36) we find, using $\sigma' = \mathcal{SF}\mathcal{T}\sigma$, that

$$\beta_{\Pi'}(\mathsf{Mod}^{\mathsf{Eq}}(\mathcal{SF}\mathcal{T}\sigma)(A)) = \beta_{\Pi'}(\mathsf{Mod}^{\mathsf{Eq}}(\sigma')(A)) = \beta_{\Pi'}(B') = B''.$$

The intuitive definition of $\mathcal{F}$ gives a $\sigma' = (\mathfrak{s}', \mathfrak{o}')$ such that

$$\mathfrak{o}'(\omega\colon s^n \to s) = \sigma(\omega)\colon s^n \to s$$

and $\mathfrak{s}'(s) = s$. Together with $\beta_{\Pi'}$, it gives $A(s) = B'(s) = B''$ and $A(\omega) = B'(\sigma(\omega)) = B''_{\sigma(\omega)}$, respectively. That is, $A'' = B''$ and we have therefore shown that (36) holds and by extension shown that β is a natural transformation.

Recall Example 3.2.1 where the entailment system of an institution was presented. There $\mathcal{I}^+$ denoted the entailment system constructed from the institution $\mathcal{I}$ and, as shown by Meseguer [105], it is also possible to expand this notion to establish a connection between maps of institutions and a map of their corresponding entailment systems. In other words, if $(\Phi, \alpha, \beta)\colon \mathcal{I}_1 \longrightarrow \mathcal{I}_2$ is a map between the institutions $\mathcal{I}_1$ and $\mathcal{I}_2$. Then $(\Phi, \alpha)\colon \mathcal{I}_1^+ \longrightarrow \mathcal{I}_2^+$ is a map of the entailment systems constructed as per Example 3.2.1.

Just as in the case of maps of entailment systems, we have to define how maps of institutions are composed. Other than

the fact that β maps model functors in reverse, no surprises are introduced.

Definition 3.6.2. Let

$$(\Phi, \alpha, \beta) \colon \mathcal{I}_1 \longrightarrow \mathcal{I}_2 \quad \text{and} \quad (\Phi', \alpha', \beta') \colon \mathcal{I}_2 \longrightarrow \mathcal{I}_3$$

be maps of institutions, then their composition

$$(\Phi', \alpha', \beta') \circ (\Phi, \alpha, \beta) \colon \mathcal{I}_1 \longrightarrow \mathcal{I}_3$$

is such that

$$(\Phi', \alpha', \beta') \circ (\Phi, \alpha, \beta) = (\Phi' \circ \Phi, \alpha'_{\mathcal{S}\Phi\mathcal{T}} \circ \alpha, \beta \circ \beta'_{\mathcal{S}\Phi\mathcal{T}}). \quad (37)$$

Proposition 3.6.1. *The composition of institutions as given in Equation 3.6.2 is itself a map of institutions.*

Proof. Assume

$$(\Phi, \alpha, \beta) \colon \mathcal{I}_1 \longrightarrow \mathcal{I}_2 \quad \text{and} \quad (\Phi', \alpha', \beta') \colon \mathcal{I}_2 \longrightarrow \mathcal{I}_3$$

are maps of institutions and let $(\Phi^\circ, \alpha^\circ, \beta^\circ)$ denote the composition

$$(\Phi', \alpha', \beta') \circ (\Phi, \alpha, \beta) \colon \mathcal{I}_1 \longrightarrow \mathcal{I}_3,$$

that is, $\Phi^\circ = \Phi' \circ \Phi$, $\alpha^\circ = \alpha'_{\mathcal{S}\Phi\mathcal{T}} \circ \alpha$, and $\beta^\circ = \beta \circ \beta'_{\mathcal{S}\Phi\mathcal{T}}$.

That the composed functor Φ° is α°-sensible is shown much like in Proposition 3.5.2 so that part will be omitted. Instead, we will concentrate on proving that the composed map fulfill the condition given in (34), that is, we wish to show that

$$M'' \models^3_{\Sigma_3} \alpha^\circ_{\Sigma_1}(\varphi) \iff \beta^\circ_{\Sigma_1}(M'') \models^1_{\Sigma_1} \varphi$$

holds for each $M'' \in \mathsf{Mod}_3(\Sigma_3)$, $\Sigma_1 \in \mathrm{Ob}(\mathtt{Sign})$, and $\varphi \in \mathsf{Sen}(\Sigma)$ with $\Sigma_2 = \mathcal{S}\Phi\mathcal{T}(\Sigma_1)$ and $\Sigma_3 = \mathcal{S}\Phi'\mathcal{T}(\Sigma_2)$. To show this is easy since we know that (Φ, α, β) and (Φ', α', β') are maps of institutions. Thus, the derivation

$$\begin{aligned} M'' \models^3_{\Sigma_3} \alpha^\circ_\Sigma(\varphi) &\iff M'' \models^3_{\Sigma_3} \alpha_{\Sigma_2}(\alpha_\Sigma(\varphi)) \\ &\iff \beta'_{\Sigma_2}(M'') \models^2_{\Sigma_2} \alpha_\Sigma(\varphi) \\ &\iff \beta_{\Sigma_1}(\beta'_{\Sigma_2}(M'')) \models^1_{\Sigma_1} \varphi \\ &\iff \beta^\circ_{\Sigma_1}(M'') \models^1_{\Sigma_1} \varphi \end{aligned}$$

establishes the validity of Proposition 3.6.1. □

Just as in the composition of maps of entailment systems, associativity follows directly from the properties of functors and natural transformations. All necessary groundwork has therefore been performed and it is now possible to define the category of institutions in a simple manner.

Definition 3.6.3. Let `Inst` denote the category of institutions. Its objects are all institutions as presented in Definition 3.1.1 and its morphisms are maps of institutions as presented in Equation 3.6.1.

As is remarked in Meseguer [105], this is not the only possible way to define this category. For example, Goguen and Burstall [71] presents an alternative, but very similar, construction for institution morphisms is presented. The one given here follows Meseguer's description and as he states: "In this presentation, I have favored the notion of a map of institutions because it fits well many natural examples and permits flexible ways of mapping theories."

3.7 THE CATEGORY OF LOGICS

Very similar to how a logic was the combination of an institution and an entailment system, a map of logics is a combination of a map of institutions and a map of entailment systems. This may be formally described in the following, simple, way.

Definition 3.7.1. Let

$$\mathcal{L}_1 = (\mathtt{Sign}_1, \mathtt{Sen}_1, \mathtt{Mod}_1, \vdash^1, \models^1) \quad \text{and}$$
$$\mathcal{L}_2 = (\mathtt{Sign}_2, \mathtt{Sen}_2, \mathtt{Mod}_2, \vdash^2, \models^2)$$

be two logics, then a *map of logics* is a triple $(\Phi, \alpha, \beta) \colon \mathcal{L}_1 \longrightarrow \mathcal{L}_2$ such that

$$(\Phi, \alpha, \beta) \colon (\mathtt{Sign}_1, \mathtt{Sen}_1, \mathtt{Mod}_1, \models^1) \longrightarrow (\mathtt{Sign}_2, \mathtt{Sen}_2, \mathtt{Mod}_2, \models^2)$$

is a map of the underlying institutions, and

$$(\Phi, \alpha) \colon (\mathtt{Sign}_1, \mathtt{Sen}_1, \vdash^1) \longrightarrow (\mathtt{Sign}_2, \mathtt{Sen}_2, \vdash^2)$$

is a map of the underlying entailment systems.

As a further example, the interested reader may wish to confirm that the institution maps of Example 3.6.1 and Example 3.6.2 actually may be expanded to maps of logics. Finally, to conclude the section, we will of course define the category of logics.

Definition 3.7.2. Let `Log` denote the category of logics. Its objects are all logics as presented in Definition 3.3.1 and it has as morphisms maps of logics as they were given in Definition 3.7.1.

3.8 PROOF CALCULI AND LOGICAL SYSTEMS

As previously mentioned, we describe the proof structure, or proof calculi, separately from syntactic implication. The reasons for this abstraction will be clarified in this section and from it, a pleasantly general definition of proof calculi will be derived. Making a long story short, the underlying reason for not grafting a proof calculi onto the entailment relation directly is that doing so would necessarily bind us to that particular proof calculi. This is a limitation since there may exist several different proof calculi that yield the same logic. Further, these proof calculi may have properties that make a calculi ideal in one situation but less suited for some other situation. A logic well known for having multiple calculi is first order logic, where natural deduction, several other so-called sequent styled calculi, and the Hilbert styled calculi give precisely the same logic but whose proof process differ in crucial details. And these are only a few examples, several other proof calculi are possible for first order logic [110].

The advantage of separating entailment and proof calculi is therefore evident; we are able to choose the proof process most advantageous for our particular purpose. Once a proof calculi has been chosen and combined with a logic, the resulting structure is known as a *logical system*.

The definition that follows is rather abstract but the strength of this is that it allows the encoding of a wide range of different notions of proof calculi. The negative consequence is that it becomes harder to grasp the underlying concept, which is not as

complex as it might seem. To help make these underlying concepts more apparent a detailed example of proof calculi will be given for equational logic.

Definition 3.8.1. A proof calculus is a tuple

$$\mathcal{P} = (\mathcal{E}, \mathsf{Pt}, \mathsf{Pr}, \pi)$$

such that

(i) $\mathcal{E} = (\mathtt{Sign}, \mathsf{Sen}, \vdash)$ is an entailment system;

(ii) $\mathsf{Pt}\colon \widehat{\mathsf{Th}}(\mathcal{E}) \longrightarrow \mathtt{Struct}$ is a functor taking theories to *proof-theoretic structures*;

(iii) $\mathsf{Pr}\colon \mathtt{Struct} \longrightarrow \mathtt{Set}$ is a functor such that for a theory $\mathsf{T} \in \mathrm{Ob}(\widehat{\mathsf{Th}}(\mathcal{P}))$, the set $\mathsf{PrPt}(\mathsf{T})$ is the *set of proofs for* T; and

(iv) $\pi\colon \mathsf{PrPt} \longrightarrow \mathsf{Sen}\mathcal{S}$ is a natural transformation such that the image of the morphism $\pi_{\mathsf{T}}\colon \mathsf{PrPt}(\mathsf{T}) \longrightarrow \mathsf{Sen}\mathcal{S}(\mathsf{T})$, for some theory $\mathsf{T} = (\Sigma, \Gamma)$, is the set $\Gamma^{\bullet}$.

Clearly, a key element for abstract nature of proof calculi is the introduction of yet another category, `Struct`, which is the category of proof-theoretic structures. It is well worth mentioning that the lack of detail about the actual structure of this category is not a mistake. In fact, this is the key to the abstract power of this categorized notion of proof calculi.

As previously mentioned, when a logic is given a specific proof calculus the combined result is known as a logical system. The definition is straight-forward.

Definition 3.8.2. A logical system, $\mathcal{S}$, is an 8-tuple

$$\mathcal{S} = (\mathtt{Sign}, \mathsf{Sen}, \mathsf{Mod}, \vdash, \models, \mathsf{Pt}, \mathsf{Pr}, \pi)$$

where

(i) $(\mathtt{Sign}, \mathsf{Sen}, \mathsf{Mod}, \vdash, \models)$ is a logic and

(ii) $(\mathtt{Sign}, \mathsf{Sen}, \vdash, \mathsf{Pt}, \mathsf{Pr}, \pi)$ is a proof calculus.

While a proof calculus by itself is not directly suited for efficient implementation it does, however, provide the basis for more efficient representations. In particular the notion of proof calculi may be extended to a structure known as *effective proof subcalculi*, this structure is described and used in Meseguer [105] as a foundation in the formalization of logic programming. Effective proof subcalculi will not be covered in this monograph but in short it adds to a logic certain restrictions on for example the axioms the logic may possess and the conclusions one may prove from these axioms. Of course, these limitations should fulfill certain requirements in order to ensure that the power of the calculi is not compromised to the point where its usefulness vanishes.

Closing the topic of proof calculi and logical systems, a detailed example of concrete proof calculi will be given. As before, the logic chosen for the example is the familiar many-sorted equational logic. In particular, a proof calculi will be constructed on the basis of so-called equational deduction.

Before tackling the examples, we will first informally recall the notion of *inference rules* as well as how they are applied – for a more thorough treatment of these notions please see, e. g., Mordechai [110]. An inference rule for some signature Σ and sentence functor Sen is simply a subset of $(\mathsf{Sen}(\Sigma))^n \times \mathsf{Sen}(\Sigma)$ where $n \in \mathbb{N}$. Consider for example the inference rule

$$\{((\varphi_1, \ldots, \varphi_n), \varphi) \mid \text{conditions}\}$$

or as it is typically written;

$$\frac{\varphi_1 \quad \cdots \quad \varphi_n}{\varphi} \quad \text{conditions.} \tag{38}$$

Informally, the inference rule of (38) states that, given a theory $T = (\Sigma, \Gamma)$, if the entailment relation is such that

$$\Gamma \vdash_\Sigma \varphi_1, \ldots, \Gamma \vdash_\Sigma \varphi_n$$

then we may, provided the given conditions are satisfied, draw the conclusion that also $\Gamma \vdash_\Sigma \varphi$ hold. In the special case where $n = 0$ we simply have the inference rule $((), \varphi)$ for some $\varphi \in \mathsf{Sen}(\Sigma)$. Of course, this rule may also be written

$$\frac{}{\varphi} \quad \text{conditions} \tag{39}$$

in the commonly used notation. Note that for an empty inference rule, such as in (39), the sentence φ will always hold and in essence, this causes φ to become a concrete axiom in the logic, one that is specific for the particular proof calculi to which it belongs.

3.8.1 *Proof Calculi for Equational Logic*

We are now able to describe one possible proof calculus for equational logic. As mentioned, this calculus will be based on so-called equational deduction that here will be somewhat extended in order to manage the added difficulty of many-sorted signatures. For a thorough introduction to these notions one may, e. g., refer to Kirchner and Kirchner [89]. First we need to establish what equational deduction is: Equational deduction is a rather uncomplicated means of determining syntactic implication using five inference rules. These are, for each equational signature $\Sigma = (S, \Omega)$ with associated families of variables X_S and Y_S, as follows:

$$\frac{}{(\forall X_S)\ t = t} \quad t \in T_\Sigma(X_S) \tag{40}$$

that informally states that a term is always equal to itself, in other words, equality is reflexive – note that this rule will also act as an axiom in the calculus;

$$\frac{(\forall X_S)\ t = u}{(\forall X_S)\ u = t} \quad t, u \in T_\Sigma(X_S), \tag{41}$$

that is, equality is symmetric;

$$\frac{(\forall X_S)\ t = u,\ (\forall X_S)\ u = v}{(\forall X_S)\ t = v} \quad t, u, v \in T_\Sigma(X_S), \tag{42}$$

that is, equality is transitive;

$$\frac{(\forall X_S)\ t = u}{(\forall Y_S)\ t\tau = u\tau} \quad t, u \in T_\Sigma(X_S), \tau \colon X_S \longrightarrow T_\Sigma(Y_S), \tag{43}$$

where τ is a substitution – we say that equality has the *substitution property*; and finally

$$\frac{(\forall X_S)\ t_1 = t'_1, \ldots, (\forall X_S)\ t_n = t'_n}{(\forall X_S)\ u\sigma = u\tau} \quad \begin{array}{l} t_1, \ldots, t_n, t'_1, \ldots, t'_n \in T_\Sigma(X_S), \\ n \geqslant 1, u \in T_\Sigma(Y_S), \text{ and} \\ \sigma, \tau \colon X_S \longrightarrow T_\Sigma(Y_S) \text{ where either} \\ \sigma(y) = \tau(y) \text{ or } \sigma(y) = t_i, \tau(y) = t'_i \\ \text{for all } y \in Y, 1 \leqslant i \leqslant n \end{array} \tag{44}$$

that states that equality uphold the so-called *congruence property*. These inference rules are fairly straight-forward. Even (44), which informally states that substituting variables by equivalent terms is acceptable.

Using this notion of equational deduction, we shall now construct the sought after proof calculus that will be denoted

$$\mathcal{P}^{Eq} = (\mathtt{Sign}^{Eq}, \mathtt{Sen}^{Eq}, \vdash^{Eq}, \mathtt{Pt}^{Eq}, \mathtt{Pr}^{Eq}, \pi^{Eq}).$$

In order for the definition to be general with regards to signature, we simply state that $(\mathtt{Sign}^{Eq}, \mathtt{Sen}^{Eq}, \vdash^{Eq})$ represents *some* equational entailment system, that is, the specifics of the $\vdash^{Eq}$-relation is left undefined. It remains to show the appearance of $\mathtt{Pt}^{Eq}$, $\mathtt{Pr}^{Eq}$, and π^{Eq}. First, let $\mathtt{Pt}^{Eq} \colon \widehat{\mathtt{Th}}(\mathcal{P}^{Eq}) \longrightarrow \mathtt{Cat}$ be a functor taking a functor $T = (\Sigma, \Gamma)$ with associated family of Σ-variables X_S to the category $\mathtt{Pt}^{Eq}(T)$ having

$$\mathrm{Ob}(P^{Eq}(T)) = T_\Sigma(X_S)$$

and for each $t, t' \in T_\Sigma(X_S)$,

$$\mathrm{Hom}_{\mathtt{Pt}^{Eq}(T)}(t, t') = \{\varphi_1, \ldots, \varphi_n \mid n \geqslant 1 \text{ and } \varphi_1, \ldots, \varphi_n \text{ is a } \textit{derivation sequence} \text{ with } \varphi_n = ((\forall X)\ t = t')\}. \tag{45}$$

By $\varphi_1, \ldots, \varphi_n$ being a derivation sequence is meant that for each φ_i, $1 \leqslant i \leqslant n$, it is the case that either $\varphi_i \in \Gamma$ or there exists an

inference rule having as an element $((\varphi_{j_1}, \ldots, \varphi_{j_m}), \varphi_i)$ where $j_1, \ldots, j_m < i$ and $m \geqslant 0$. The following brief example illustrates a concrete morphism in a category constructed by the $\mathsf{Pt}^{\mathrm{Eq}}$ functor.

It is easy to convince oneself that morphism composition in this category is nothing more than concatenation of the derivation sequences. Note that the inference rules of (40), (41), and (42) are represented directly in the structure of this category. That is, the reflexivity rule is represented by the identity morphisms. The symmetry rule states that all morphisms in $\mathsf{Pt}^{\mathrm{Eq}}(T)$, for some theory T, are isomorphic. Finally, the transitivity rule is directly represented by morphism composition.

Having specified the $\mathsf{Pt}^{\mathrm{Eq}}$ functor, the $\mathsf{Pr}^{\mathrm{Eq}}\colon \mathtt{Cat} \longrightarrow \mathtt{Set}$ functor is defined simply

$$\begin{aligned}&\mathsf{Pr}^{\mathrm{Eq}}(\mathsf{Pt}^{\mathrm{Eq}}(T)) = \\ &\quad \{(X_S, t, p, t') \mid p \in \mathrm{Hom}_{\mathsf{Pt}^{\mathrm{Eq}}(T)}(t, t'), t, t' \in T_\Sigma(X_S)\} \end{aligned} \tag{46}$$

for each theory $T = (\Sigma, \Gamma)$ having associated family of variables X. Informally, this states that $\mathsf{Pr}^{\mathrm{Eq}}\mathsf{Pt}^{\mathrm{Eq}}T$ is the set of all equations provable from the theory T including their proof. Subsequently, $\pi^{\mathrm{Eq}}\colon \mathsf{Pr}^{\mathrm{Eq}} \circ \mathsf{Pt}^{\mathrm{Eq}} \longrightarrow \mathsf{Sen}^{\mathrm{Eq}} \circ \mathcal{S}$ simply projects the equation from these triples, that is,

$$\pi^{\mathrm{Eq}}_T(X_S, t, p, t') = ((\forall X_S)\ t = t')$$

for each $(X_S, t, p, t') \in \mathrm{Ob}(\mathsf{Pr}^{\mathrm{Eq}}\mathsf{Pt}^{\mathrm{Eq}}T)$. In other words, π^{Eq}, forgets the proof and only remembers the proven equation. Clearly, it is required to establish that π^{Eq} really is a natural transformation, this is done in the following proposition.

Proposition 3.8.1. *Let $\pi^{Eq}\colon \mathsf{Pr}^{Eq}\mathsf{Pt}^{Eq} \longrightarrow \mathsf{Sen}^{Eq}\mathcal{S}$ be as given above, then π^{Eq} is a natural transformation.*

Proof. For simplicity, let in this proof $\mathsf{Sen} = \mathsf{Sen}^{\mathrm{Eq}}$, $\mathsf{Pt} = \mathsf{Pt}^{\mathrm{Eq}}$, $\mathsf{Pr} = \mathsf{Pr}^{\mathrm{Eq}}$, and $\pi = \pi^{\mathrm{Eq}}$. Then following the usual procedure, to

show that π is a natural transformation, it is required to show that the diagram

$$\begin{array}{ccc} \mathsf{PrPt}\mathsf{T} & \xrightarrow{\pi_{\mathsf{T}}} & \mathsf{Sen}\mathcal{S}\mathsf{T} \\ {\scriptstyle \mathsf{PrPt}\sigma_{\mathsf{Th}}}\downarrow & & \downarrow{\scriptstyle \mathsf{Sen}\mathcal{S}\sigma_{\mathsf{Th}}} \\ \mathsf{PrPt}\mathsf{U} & \xrightarrow[\pi_{\mathsf{U}}]{} & \mathsf{Sen}\mathcal{S}\mathsf{U} \end{array}$$

commutes for each theorem morphism $\sigma_{\mathsf{Th}} \colon \mathsf{T} \longrightarrow \mathsf{U}$, that is,

$$\mathsf{Sen}\mathcal{S}\sigma_{\mathsf{Th}} \circ \pi = \pi \circ \mathsf{PrPt}\sigma_{\mathsf{Th}}.$$

This will be done by showing that

$$(\mathsf{Sen}\mathcal{S}\sigma_{\mathsf{Th}} \circ \pi_{\mathsf{T}})(x) = (\pi_{\mathsf{U}} \circ \mathsf{PrPt}\sigma_{\mathsf{Th}})(x) \tag{47}$$

for some $x = (X_S, t, p, t') \in \mathsf{PrPT}$ and theory $\mathsf{T} = (\Sigma, \Gamma)$. The left-hand side of (47) is then

$$\begin{aligned} (\mathsf{Sen}\mathcal{S}\sigma_{\mathsf{Th}} \circ \pi_{\mathsf{T}})(x) &= \mathsf{Sen}\mathcal{S}\sigma_{\mathsf{Th}}(\pi_{\mathsf{T}}(X_S, t, p, t')) \\ &= \mathsf{Sen}\mathcal{S}\sigma_{\mathsf{Th}}((\forall X_S)\ t = t') \\ &= ((\forall \sigma(X_S))\ \widehat{\sigma}(t) = \widehat{\sigma}(t')) \end{aligned}$$

while the right-hand side of (47) is

$$\begin{aligned} (\pi_{\mathsf{U}} \circ \mathsf{PrPt}\sigma_{\mathsf{Th}})(x) &= \pi_{\mathsf{U}}(\mathsf{PrPt}\sigma_{\mathsf{Th}}(X_S, t, p, t')) \\ &= \pi_{\mathsf{U}}(\sigma(X_S), \widehat{\sigma}(t), p', \widehat{\sigma}(t'))) \\ &= ((\forall \sigma(X_S))\ \widehat{\sigma}(t) = \widehat{\sigma}(t')) \end{aligned}$$

Where p' is the proof of $(\forall \sigma(X_S))\ \widehat{\sigma}(t) = \widehat{\sigma}(t')$ in the U theory. We know that there exists such a proof due to the construction of σ_{Th}, see Definition 3.4.2. Clearly, both the left-hand side and the right-hand side yield the same equation and we may subsequently conclude that π is a natural transformation. □

The proof process that $\mathsf{Pt}^{\mathrm{Eq}}$ and $\mathsf{Pr}^{\mathrm{Eq}}$ perform is both sound and complete – see, e. g., Loeckx et al. [98] for formal proof – and as a result, it is clear from its construction that the image of π_{T} is the set of equations provable from the theory $\mathsf{T} = (\Sigma, \Gamma)$. In other words, the image contains exactly the sentences of $\Gamma^{\bullet}$. Just as required in condition 3.8.1(iv).

3.9 THE CATEGORIES OF PROOF CALCULI AND LOGIC SYSTEMS

The two major concepts left to describe are the extensions of proof calculi and logic systems into their respective category. As before, it is first necessary to establish precisely what constitutes a mapping in the context of proof calculi and logic systems. Fortunately, once a mapping of proof calculi has been described, the logic system mapping will follow without much effort.

Recall that a proof calculi consists in part of an entailment system, a map of proof calculi will subsequently consist in part of a map of entailment systems. Further, a map of proof calculi will also be required to translate the proofs of the source proof calculi into valid proofs in the target calculi. These notions are formalized in Definition 3.9.1 below.

Definition 3.9.1. Let

$$\mathcal{P}_1 = (\mathsf{Sign}_1, \mathsf{Sen}_1, \vdash^1, \mathsf{Pt}_1, \mathsf{Pr}_1, \pi^1) \text{ and}$$
$$\mathcal{P}_2 = (\mathsf{Sign}_2, \mathsf{Sen}_2, \vdash^2, \mathsf{Pt}_2, \mathsf{Pr}_2, \pi^2)$$

be two proof calculi. Then a *map of proof calculi*

$$(\Phi, \alpha, \gamma) \colon \mathcal{P}_1 \longrightarrow \mathcal{P}_2$$

is such that

$$(\Phi, \alpha) \colon (\mathsf{Sign}_1, \mathsf{Sen}_1, \vdash^1) \longrightarrow (\mathsf{Sign}_2, \mathsf{Sen}_2, \vdash^2)$$

maps the embedded entailment system of $\mathcal{P}_1$ to the one of $\mathcal{P}_2$, and

$$\gamma \colon \mathsf{Pr}_1 \circ \mathsf{Pt}_1 \longrightarrow \mathsf{Pr}_2 \circ \mathsf{Pt}_2 \circ \Phi$$

is a natural transformation such that

$$\alpha\mathcal{S} \circ \pi^1 = \pi^2 \Phi \circ \gamma. \tag{48}$$

Informally, one may say that (48) requires the map to be such that mapping the projected proofs of the source logic to sentences

of the target logic yield the same result as mapping the proofs first and then projecting the sentences.

Clearly, one might ask whether it is possible to construct a map when one is not fortunate enough to have a target proof calculi with a deductive system that subsumes the deductive system of the source proof calculi. The answer may be found by considering (48). Under the assumption that the source and target proof calculi have been fixed, we are able to alter only α and γ – the two natural transformations π^1 and π^2 are part of the proof calculi. Therefore, since we are unable to translate proofs without loss of information, we will have construct α and γ such that this loss of information does not introduce inconsistencies. By constructing these two in a clever manner, it is indeed possible to construct maps from proof calculi of greater expressive power to less powerful ones. Put shortly, this situation is similar to the one presented in Example 3.6.1 where the institution of many-sorted equational logic was taken to its unsorted same. In both cases, some loss of information is unavoidable in the general case but this is sometimes an acceptable price to pay.

Composition of proof calculi is defined precisely in the manner one might expect. That is, the Φ and α components are composed in the manner of maps of entailment system. Finally, the γ components are vertically composed – with care taken to not forget the stacked Φ component. The following more formal description illustrate this.

Definition 3.9.2. Let

$$\begin{aligned}(\Phi, \alpha, \gamma) &\colon \mathcal{P}_1 \longrightarrow \mathcal{P}_2\\ (\Phi', \alpha', \gamma') &\colon \mathcal{P}_2 \longrightarrow \mathcal{P}_3\end{aligned}$$

be maps of proof calculi, then the composition of them,

$$(\Phi, \alpha, \gamma) \circ (\Phi', \alpha', \gamma') \colon \mathcal{P}_1 \longrightarrow \mathcal{P}_3,$$

is such that

$$(\Phi, \alpha, \gamma) \circ (\Phi', \alpha', \gamma') = (\Phi' \circ \Phi, \alpha'_{\mathcal{S}\Phi\mathcal{T}} \circ \alpha, \gamma'_{\Phi} \circ \gamma)$$

That composition of maps of proof calculi as given in Definition 3.9.2 yield a map of proof calculi and that composition is associative is shown much like in the previous maps so this will not be formulated in a proposition here. Instead, we simply conclude that – taking advantage of the results presented in this section – the category of proof calculi is defined as follows.

Definition 3.9.3. Let `PCalc` denote the category of proof calculi. Its objects are all proof calculi as described in Definition 3.8.1 and its morphisms are maps of proof calculi, given in Definition 3.9.1.

Extending this definition to incorporate logic systems give us another category; the one for logical systems.

Definition 3.9.4. Let `LogSys` denote the category of logical systems having as objects all logical systems as per Definition 3.8.2 and having as morphisms maps such that if $\mathcal{S}_1$ and $\mathcal{S}_2$ are logical systems, then a map between them is a quadruple

$$(\Phi, \alpha, \beta, \gamma) \colon \mathcal{S}_1 \longrightarrow \mathcal{S}_2$$

such that

$$(\Phi, \alpha, \beta) \colon \operatorname{logic}(\mathcal{S}_1) \longrightarrow \operatorname{logic}(\mathcal{S}'_2)$$

maps the logic component of $\mathcal{S}_1$ to its counterpart in $\mathcal{S}_2$ and

$$(\Phi, \alpha, \gamma) \colon \operatorname{pcalc}(\mathcal{S}_1) \longrightarrow \operatorname{pcalc}(\mathcal{S}_2)$$

similarly maps the proof calculus of $\mathcal{S}_1$ to the proof calculus of $\mathcal{S}_2$. Here, logic and pcalc are projection operations having the obvious definition – these have been introduced merely for notational convenience.

3.10 FRAMEWORK LOGICS

This chapter has first and foremost shown that we may represent logics and logical systems in the categorical context. The primary strength of these representations is that one may in a rather convenient manner move from one representation to another. The running example having been the transition between equational

logic and first order logic. While the construction of the various morphisms in this example has been relatively straight-forward, this may not always be the case. In fact, as one might imagine, they can be considerably harder. This problem is further compounded by the continuous creation of new logics as research into new areas often give rise to the need of new representations and notions of logic. For each new logic created, there exists a need to identify and construct mappings from the new logic to the old familiar ones; certainly not a trivial problem.

These difficulties form to a large part the motivation behind the notion of a *framework logic*. That is, very general style of logic in which many other logics may be represented. This allows the framework logic to act as a central hub when translating between logics, that is, if one with to go from a newly created logic to various other logics it is possible to first translate it into the framework logic and then use one of the – hopefully existing – translation from the framework logic to the target logics. The benefit is obvious, we save ourselves a lot of work by only having to specify a small number of translations. The problem is also obvious, a framework logic would have to be very general indeed before it is practically useful. Ideally, one would like the framework logic to be general enough to represent most logics of interest – known and unknown – while at the same time being manageable in size and complexity. A few candidates have been proposed [115] and one that has received quite some attention is *rewriting logic*, which was introduced by Martí-Oliet and Meseguer [104].

3.11 REFINEMENT OF CLINICAL EVIDENCE

This section outlines a more elaborate example of general logics applied in the medical field. The text here a somewhat elaborated version of results presented in Eklund et al. [50].

Diagnosis of cognitive disorder differentiates between cognitive impairment, delirium and dementia. Similarly, in the case of dementia, differentiation is required in further steps when diagnosing dementia of Alzheimer's type and various non-Alzheimer dementia types such as vascular dementia. In these further steps

neurological functions, e. g., as determined by radiological signs become important parts of the diagnosing process. It is also also important to note how dementia may connect with a number of general medical conditions, and dementia is generally also classified with respect to its severity.

In this view we should note that diagnosis is a prerequisite for intervention that may be based, e. g., on organic, personal and social aspects. In an organic view, interventions are mostly pharmacological with improvement of neurotransmissions or avoidance of neuronal degeneration as typical goals. At a personal level objectives are to maintain or even improve capabilities with respect to daily living, thus enabling elderly to remain in their own homes. Personal and social levels of intervention are performed by different professional groups as compared to corresponding interventions at organic level. Nevertheless, the overall resources of information is the same albeit used at different levels with different degrees of specificity. In this holistic view rule base representation is quite challenging as the requirement is that underlying knowledge and well-foundedness of information for diagnosis and treatment must be concise and sufficient also for smooth transfer between professional domains.

Diagnosis of cognitive disorder is initially based on observations concerning memory functions such as progressing difficulties with episodic and semantic memory. Representation of these 'measurements' can be done with different levels of specificity and confidence. At each representation level, these rules must be 'correct' is some sense with respect to the given evidence-based guidelines. However, even more important is correctness preservation when shifting from one representation level to another. Concerning this correctness preservation, general logics provides the appropriate formal framework for management transformations between logics and logical representations.

As we've seen, general logics provides an appropriate framework for representing these refinements of information. As will become evident, a logic described within this categorical framework contains suitable building blocks, e. g., in form of functors and natural transformations, that enables representation of information at various user levels. General logics has previously not

been used for representation of clinical guidelines, and therefore we need some further categorical developments to support our developments. A typical basic question in this respect is whether to analyze rules within one and the same entailment system involving a category of signatures or to split the entailment system into different entailment systems based on separating signatures of interest. We adopt the latter and establish the necessary theoretical framework for this approach.

3.11.1 *Splitting Entailment Systems*

We may in an intuitive sense split an entailment system into two separate entailment systems by choosing appropriate signature morphisms. This split will also give rise to a map between the two newly constructed entailment systems. The following proposition makes this claim precise.

Proposition 3.11.1. *Let* $\mathcal{E} = (\mathsf{Sign}, \mathsf{Sen}, \vdash)$ *be an entailment system. Additionally, let*

$$\mathcal{E}_1 = (\mathsf{Sign}_1, \mathsf{Sen}_1, \vdash^1) \quad \textit{and} \quad \mathcal{E}_2 = (\mathsf{Sign}_2, \mathsf{Sen}_2, \vdash^2)$$

be such that Sign_1 *and* Sign_2 *are subcategories of* Sign *and* Sen_1, Sen_2, $\vdash^1$, *and* $\vdash^2$ *are the restrictions of* Sen *and* $\vdash$ *to the signatures of their respective signature categories. Further, place on* Sign_1 *and* Sign_2 *the restriction that they may contain only identity morphisms and, finally, for each* $\Sigma_1 \in \mathrm{Ob}(\mathsf{Sign}_1)$, *choose a signature morphism* $\sigma \in \mathrm{Hom}_{\mathsf{Sign}}(\Sigma_1, \Sigma_2)$, $\Sigma_2 \in \mathrm{Ob}(\mathsf{Sign}_2)$.

Under these conditions $\mathcal{E}_1$ *and* $\mathcal{E}_2$ *are entailment systems and*

$$(\Phi, \alpha) \colon \mathcal{E}_1 \longrightarrow \mathcal{E}_2$$

as given below is a map between them. The functor $\Phi \colon \widehat{\mathsf{Th}}(\mathcal{E}_1) \longrightarrow \widehat{\mathsf{Th}}(\mathcal{E}_2)$ *takes* $\mathcal{E}_1$*-theories to* $\mathcal{E}_2$*-theories and is defined for each* $\mathcal{E}_1$*-theory* (Σ, Γ) *by*

$$\begin{aligned} \Phi(\Sigma, \Gamma) &= (\sigma(\Sigma), \mathsf{PSen}(\sigma)(\Gamma)) \\ \Phi(\mathrm{id}_{(\Sigma,\Gamma)}) &= \mathrm{id}_{\Phi(\Sigma,\Gamma)} \end{aligned}$$

and $\alpha \colon \mathsf{Sen}_1 \longrightarrow \mathsf{Sen}_2 \circ \mathcal{S} \circ \Phi \circ \mathcal{T}$ *is a natural transformation similarly defined by* $\alpha_\Sigma = \mathsf{Sen}(\sigma)$.

Proof. First we observe that since a known entailment system has been restricted to a subset of its signatures and the signature morphisms have at the same time been limited to only the identity morphisms, it is trivially true that $\mathcal{E}_1$ and $\mathcal{E}_2$ are entailment systems. It remains to show that (Φ, α) is a map from $\mathcal{E}_1$ to $\mathcal{E}_2$.

Second, because $\mathtt{Sign}_1$ and $\mathtt{Sign}_2$ only contain identity morphisms, Φ is by construction a functor and likewise α is a natural transformation since

$$\alpha_\Sigma \circ \mathsf{Sen}_1(\mathrm{id}_\Sigma) = (\mathsf{Sen}_2 \circ \mathcal{S} \circ \Phi \circ \mathcal{T})(\mathrm{id}_\Sigma) \circ \alpha_\Sigma$$

for all $\Sigma \in \mathrm{Ob}(\mathtt{Sign}_1)$. We now have that condition 3.5.1(i) holds since for each $\Sigma \in \mathrm{Ob}(\mathtt{Sign}_1)$ and $\Gamma, \Gamma' \in \mathsf{PSen}_1(\Sigma)$ it is the case that

$$\mathcal{S}\Phi(\Sigma, \Gamma) = \sigma(\Sigma) = \mathcal{S}\Phi\mathcal{T}\mathcal{S}(\Sigma, \Gamma').$$

Condition 3.5.1(ii) holds since

$$\begin{aligned}(\mathcal{A}\Phi(\Sigma, \Gamma))^\bullet &= (\mathsf{PSen}(\sigma)(\Gamma))^\bullet \\ &= (\mathsf{PSen}(\sigma)(\emptyset) \cup \mathsf{PSen}(\sigma)(\Gamma))^\bullet \\ &= (\mathcal{A}\Phi(\Sigma, \emptyset) \cup \mathsf{P}\alpha_\Sigma(\Gamma))^\bullet.\end{aligned}$$

Finally, condition 3.5.1(iii) holds since by the definition of entailment systems we know that

$$\Gamma \vdash_\Sigma \varphi \implies \mathsf{PSen}(\sigma)(\Gamma) \vdash_{\Sigma'} \mathsf{Sen}(\sigma)(\varphi)$$

for each $\Sigma' \in \mathrm{Ob}(\mathtt{Sign})$ such that $\sigma \in \mathrm{Hom}_{\mathtt{Sign}}(\Sigma, \Sigma')$. This is generalization of the situation given by the definition of maps of entailment systems applied to (Φ, α), i. e., we have

$$\begin{aligned}\Gamma \vdash^1_\Sigma \varphi &\implies \mathsf{P}\alpha_\Sigma(\Gamma) \cup \mathcal{A}_2\Phi(\Sigma, \emptyset) \vdash^2_{\mathcal{S}_2\Phi(\Sigma,\emptyset)} \alpha_\Sigma(\varphi) \\ &\implies \mathsf{PSen}(\sigma)(\Gamma) \vdash^2_{\sigma(\Sigma)} \mathsf{Sen}(\sigma)(\varphi).\end{aligned}$$

□

3.11.2 *Cognitive Disorder Differential Diagnosis*

In this section we will consider mappings between entailment systems, where the object sets for the categories of signatures

are one-pointed. It is then natural to require that the signature categories contain only identity morphisms for their respective signatures. The justification for this split of signature categories and entailment systems is given by Proposition 3.11.1.

The mapping situation is based on the following scenarios.

Scenario 1: Granularity of knowledge

Successful pharmacological treatment of type Alzheimer's disease requires an early detection of cognitive decline. An early detection happens typically during home care, i. e., when professionals from the social and nursing area are in direct and frequent contact with the elderly. Diagnosis cannot be performed in this environment, but initial and initiating observations of symptoms and signs used in guidelines for diagnosis of dementia can be (systematically using different validated screening tools) collected and brought forward to primary care physicians. These medical experts in turn carry out preliminary diagnostic tasks before referring the patient to a regular geriatric investigation concerning the suspected state of dementia. In the process different screening-tools and guidelines are used that capture knowledge at different levels of granularity.

What are the respective information types and rule representations for these professional groups? Can we guarantee consistency when information and knowledge is mapped between ontological domains as understood and used by these professional groups?

Scenario 2: Many-valuedness of truth

During the course of diagnosis it is common that information viewed is considered not accurate enough or knowledge entailed is understood as just vaguely true. Nevertheless, clinicians may have to reach conclusions and make decisions as a basis for further intervention steps based on these premises. This means that qualifications and truth values may have to be sharpened to becoming binary.

When moving from vagueness, and even accurately estimated vagueness, to presenting decisions and rules based thereof in a

more crisp fashion, how should these transformations be specified so that information loss is minimized and the remaining crisp situation is an optimal representation of the vague situation?

Let $\mathcal{E}_i = (\mathtt{Sign}_i, \mathtt{Sen}_i, \vdash_i)$, $i = 1, 2$, be entailment systems with $\mathrm{Ob}(\mathtt{Sign}_i) = \{(S_i, \Omega_i)\}$. We will picture $\mathcal{E}_1$ as an equational logic basically using Boolean terms. In order to do this let

$$
\begin{aligned}
S_1 &= \{\mathtt{BOOL}\} \\
\Sigma_1 &= \{\mathtt{false}\colon \to \mathtt{BOOL}, \mathtt{true}\colon \to \mathtt{BOOL}, \\
&\qquad \mathtt{OR}\colon \mathtt{BOOL} \times \mathtt{BOOL} \to \mathtt{BOOL}\}.
\end{aligned}
$$

The $\mathtt{Sen}_i$ functors are specified for equational logic as in Definition 3.1.4, i. e., sentences for a signature $\Sigma = (S, \Omega)$ are triples $(\forall X_S)\ t = t'$ with X_S being a $\mathtt{Set}_S$ object and t, t' being Σ-terms.

The equations for `OR` are as usual. Part of the theory $T_1 = (\Sigma_1, \Gamma_1)$, where $\Sigma_1 = (S_1, \Omega_1)$, for classifying delirium according to DSM-IV guidelines [5] is given as

```
op false true : -> BOOL .
op OR : BOOL BOOL -> BOOL .
var ... X_episodic X_semantic X_shortterm ... : BOOL .
eqn OR ... .
eqn DEO_CogDisDSMdelirium =
  ...
  OR
    {
     X_episodic
     X_semantic
     X_shortterm
    }
  ... .
eqn X_episodic = false .
eqn X_semantic = true .
eqn X_shortterm = true .
```

in some informal specification language syntax. In this view, we should see

```
eqn OR ... .
```

```
eqn DEO_CogDisDSMdelirium =
  ...
  OR
    {
     X_episodic
     X_semantic
     X_shortterm
    }
  ... .
```

as the set of axioms, and

```
X_episodic = false .
X_semantic = true .
X_shortterm = true .
```

as specific patient data, in the end stored and retrieved from the electronic patient record.

This arrangement could in Scenario 1 be seen as the rule base used within home care and could in Scenario 2 be seen as part of the guideline needed for crisp decisions.

Entailment system $\mathcal{E}_2$ will also be equational but with more sorts and operators. We will include the sort `qualICF` to enable representation of qualification values according to ICF [136]. Let now

$$
\begin{aligned}
S_2 &= \{\texttt{BOOL}, \texttt{qualICF}\} \\
\Omega_2 &= \{\texttt{false:} \to \texttt{BOOL}, \texttt{true:} \to \texttt{BOOL}, \\
&\qquad \texttt{OR:}\ \texttt{BOOL} \times \texttt{BOOL} \to \texttt{BOOL}, \\
&\qquad \texttt{0:} \to \texttt{qualICF}, \texttt{1:} \to \texttt{qualICF}, \texttt{2:} \to \texttt{qualICF}, \\
&\qquad \texttt{3:} \to \texttt{qualICF}, \texttt{4:} \to \texttt{qualICF}, \\
&\qquad \texttt{a:}\ \texttt{qualICF} \to \texttt{BOOL}\}
\end{aligned}
$$

The operator `a` thus converts qualification values to boolean values. (Part of) theory $T_2 = (\Sigma_2, \Gamma_2)$, where $\Sigma_2 = (S_2, \Omega_2)$, becomes[2]

2 Granularity of memory types follows Tulving [134], i. e., a distinction is made between semantic and episodic memory, thus extending the ICF codes [136] in the example with an additional level of granularity.

```
op false true : -> BOOL .
op OR : BOOL BOOL -> BOOL .
op 0 1 2 3 4 : qualICF .
op a : qualICF -> BOOL .
var ... X_b14411 X_b14412 X_b1440 ... : BOOL .
eqn OR ... .
eqn a 0 = false .
eqn a 1 = true .
eqn a 2 = true .
eqn a 3 = true .
eqn a 4 = true .
eqn DEO_CogDisDSMdelirium =
  ...
  OR
    {
     a X_b14411 *** ICF code for longterm episodic memory
     a X_b14412 *** ICF code for longterm semantic memory
     a X_b1440  *** ICF code for shortterm memory
    }
  ...
eqn X_b14411 = 0 .
eqn X_b14412 = 1 .
eqn X_b1440 = 3 .
```

Thus, e. g., $\mathtt{a}(0) = \mathtt{false} \in \mathsf{Sen}_2((S_2, \Omega_2))$. Further, the signature morphism $\sigma = (\mathfrak{s}, \mathfrak{o})\colon \Sigma_1 \longrightarrow \Sigma_2$ is given by

$$\mathfrak{s}(\mathtt{BOOL}) = \mathtt{BOOL},$$

and

$$\mathfrak{o}(\mathtt{false}) = \mathtt{false} \qquad \mathfrak{o}(\mathtt{true}) = \mathtt{true} \qquad \mathfrak{o}(\mathtt{OR}) = \mathtt{OR}.$$

The entailment morphism $\mathcal{E}_1 \longrightarrow \mathcal{E}_2$ is then trivial as it basically embeds knowledge without changing granularity.

The embedding $\mathcal{E}_2 \longrightarrow \mathcal{E}_1$, on the other hand, is non-trivial. In this case we need an 'identity' $\mathtt{i}\colon \mathtt{BOOL} \to \mathtt{BOOL}$ as an operator in Ω_1, i. e., having

```
eqn i false = false .
eqn i true = true .
```

as its equations in Γ_1.

Remark. Using negation instead of `i` obviously changes the rule base completely. However, by using `i`, misinterpretations due to cultural differences in the intuitive mappings can be handled, such as situations when negations are confirmed with a positive response can be captured here, which otherwise requires extensive context information. E. g., "affected memory function? - yes" vs. "memory function? - affected".

Let the signature morphism $\sigma = (\mathfrak{s}, \mathfrak{o}) \colon \Sigma_2 \longrightarrow \Sigma_1$ be given by

$$\mathfrak{s}(\texttt{BOOL}) = \texttt{BOOL} \qquad \mathfrak{s}(\texttt{qualICF}) = \texttt{BOOL}.$$

Further, let

$$\begin{aligned} \mathfrak{o}(\texttt{false}) &= \texttt{false} & \mathfrak{o}(\texttt{true}) &= \texttt{true} \\ \mathfrak{o}(\texttt{OR}) &= \texttt{OR} & \mathfrak{o}(\texttt{0}) &= \texttt{false} \\ \mathfrak{o}(\texttt{1}) &= \texttt{true} & \mathfrak{o}(\texttt{2}) &= \texttt{true} \\ \mathfrak{o}(\texttt{3}) &= \texttt{true} & \mathfrak{o}(\texttt{4}) &= \texttt{true} \\ \mathfrak{o}(\texttt{a}) &= \texttt{i}. \end{aligned}$$

We now have, e. g.,

$$\begin{aligned} \mathsf{Sen}(\sigma)(\texttt{X\_b14411} = \texttt{0}) &= (\texttt{X\_episodic} = \texttt{false}) \\ \mathsf{Sen}(\sigma)(\texttt{a}(\texttt{X\_b14411}) = \texttt{false}) &= (\texttt{X\_episodic} = \texttt{false}) \end{aligned}$$

Qualification values could also have been converted within a many-valued logic framework, and in this case `BOOL` would be extended to capture many-valuedness. Converters must be defined accordingly. In future papers we will explore these examples in more detail, and including more complete versions of rule bases. More elaborate ingredients are then necessary, such as provided by formal descriptions in Gottwald [74], Hájek [76].

4

GENERALIZED GENERAL LOGICS

We will in this chapter consider a generalization of the general logics framework. This generalization is intended to provide a more convenient and informative framework for working with non-classical logics such as, e. g., logics having non-binary notions of truth or logics operating over uncertain information.

The construction to follow is therefore grounded on the observation that a fundamental notion of traditional logics is that of sets of sentences. These occur most prominently in the previously presented constructions of sets of axioms and logical theories. In a generalized setting – such as in the aforementioned case where representation of uncertain information is of central importance – we must therefore extend the concept of sets of sentences in a suitable way. Clearly, what is "suitable" is dependent on the type of application being considered and to accommodate a wide variety of choices in this regard we base our notion of structured sets of sentences on partially ordered monads. These provide sufficient structure to define the generalized logic structures while remaining relatively unbiased as to the choice of structuring. In the name of brevity, such structured sets of sentences will in the general case be called *theoremata*.

While theoremata in many cases are representable in traditional general logics and indeed, if the full power of ZFC is brought to bear we are not adding any expressive power by preferring the theoremata view over plain sets of sentences. For example, as shown by Diaconescu [36, Proposition 3.3], we may in a multiple-valued setting have the sentences produced by the Sen functor include a value indicating the membership of the sentence. Such construction are, however, necessarily low level, non-compositional, and generally rather uninformative as to the grander structure of non-classical logic. Thus, rather than including an explicit membership value inside each sentence in a set of

sentences we may instead wish to have sentences in a fuzzy set of sentences. By adopting different partially ordered monads

The generalizations that will be presented are not drastic in that they depart from the general appearance of the constructions of general logics. In fact, this generalization may be viewed as rather conservative and indeed all constructions and results from Chapter 3 transfer straightforwardly to the generalized setting.

4.1 INSTITUTIONS, ENTAILMENT SYSTEMS, AND LOGICS

In this section we present the generalized versions of the structures given in the previous section. To this end we will, in this section, remain fully abstract concerning the $\mathtt{Sign}$ category.

Definition 4.1.1. A generalized entailment system, $\mathcal{E}$, is a structure $\mathcal{E} = (\mathtt{Sign}, \mathtt{Sen}, \boldsymbol{\Delta}, \mathbf{L}, \vdash)$ where

- $\mathtt{Sign}$ is a category of signatures;
- $\mathtt{Sen}$ is a functor $\mathtt{Sen}\colon \mathtt{Sign} \longrightarrow \mathtt{Set}$ taking signatures to sentences;
- $\boldsymbol{\Delta} = (\Delta, \preceq, \eta, \mu)$ is a partially ordered monad over $\mathtt{Set}$ with an object of $\Delta\mathtt{Sen}(\Sigma)$ being called a *theoremata*;
- $\mathbf{L}$ is a completely distributive lattice; and
- $\vdash$ is a family of $\mathbf{L}$-valued relations consisting of

  $$\vdash_\Sigma \colon \Delta\mathtt{Sen}(\Sigma) \times \Delta\mathtt{Sen}(\Sigma) \longrightarrow \mathbf{L}$$

  for each signature $\Sigma \in \mathrm{Ob}(\mathtt{Sign})$ where $\vdash_\Sigma$ is called a Σ-*entailment*.

These are subject to the condition that, for $\Gamma_1, \Gamma_2, \Gamma_3 \in \Delta\mathtt{Sen}(\Sigma)$, each $\vdash_\Sigma$

(i) is reflexive, that is, $(\Gamma_1 \vdash_\Sigma \Gamma_1) = \top$;

(ii) is *axiom monotone*, that is,

$$((\Gamma_1 \vee \Gamma_2) \vdash_\Sigma \Gamma_3) \geqslant (\Gamma_1 \vdash_\Sigma \Gamma_3) \vee (\Gamma_2 \vdash_\Sigma \Gamma_3);$$

(iii) is *consequent invariant*, i. e.,

$$(\Gamma_1 \vdash_\Sigma \Gamma_2) \wedge (\Gamma_1 \vdash_\Sigma \Gamma_3) = (\Gamma_1 \vdash_\Sigma (\Gamma_2 \vee \Gamma_3));$$

(iv) is transitive in the sense that

$$(\Gamma_1 \vdash_\Sigma \Gamma_2) \wedge ((\Gamma_1 \vee \Gamma_2) \vdash_\Sigma \Gamma_3) \leqslant (\Gamma_1 \vdash_\Sigma \Gamma_3); \text{ and}$$

(v) is an $\vdash$-*translation*, meaning that

$$(\Gamma_1 \vdash_\Sigma \Gamma_2) \leqslant (\Delta\mathsf{Sen}(\sigma)(\Gamma_1) \vdash_{\Sigma'} \Delta\mathsf{Sen}(\sigma)(\Gamma_2))$$

for all signature morphisms $\sigma \in \mathrm{Hom}_{\mathtt{Sign}}(\Sigma, \Sigma')$.

If $\Gamma_1 \vdash_\Sigma \Gamma_2$, then we say that Γ_1 are the *axioms* and Γ_2 *is derivable from* Γ_1 or, alternatively, that Γ_2 *is a logical consequence of* Γ_1. Again, we frequently omit the subscript of $\vdash_\Sigma$ when the signature is obvious from context.

For the above definition it should be noted that the definition introduces an additional condition, 4.1.1(iii), that does not directly correspond to a condition in traditional entailment systems. The condition is added to indicate that joining two logical consequences will yield a degree of truth equal to the lesser of two consequences. The condition is not necessary in traditional entailment systems since the logical consequences are atomic and joins are not defined.

Further, note that we have departed from the traditional view in considering $\vdash$ as acting over richer structures of sentences and being L-valued, but also in the nature of the right operand. In this generalized definition we ask for a theoremata where it in the traditional setting would be a plain sentence. Clearly, however, a sentence by itself may not have an intelligible interpretation in the logic. For example, if $\mathbf{\Phi}$ is chosen to be $\mathbf{L}$ then for a sentence φ we must also have an associated degree of membership. Nevertheless, we may choose the interpretation that $\Gamma \vdash_\Sigma \varphi$ is taken to be the same as $\Gamma \vdash_\Sigma \eta_{\mathsf{Sen}(\Sigma)}(\varphi)$. We may motivate this choice as being natural in light of the following simple result.

Lemma 4.1.1. *Given* $\Gamma', \Gamma \in \Delta\mathsf{Sen}(\Sigma)$ *such that* $\Gamma' \preceq \Gamma$ *then* $(\Gamma \vdash \Gamma') = \top$.

Proof. By 4.1.1(i) we have $(\Gamma' \vdash \Gamma') = \top$. Further, from 4.1.1(ii) we find that

$$(\Gamma' \vdash \Gamma') \leqslant (\Gamma \vdash \Gamma').$$

It then immediately follows that $(\Gamma \vdash \Gamma') = \top$. □

Note, as a special case, this lemma shows, that if we have $\eta_{\mathsf{Sen}(\Sigma)}(\varphi) \preceq \Gamma$ for some sentence $\varphi \in \mathsf{Sen}(\Sigma)$, then we also have $(\Gamma \vdash \eta_{\mathsf{Sen}(\Sigma)}(\varphi)) = \top$. Thus, with the previously mentioned interpretation we have $(\Gamma \vdash \varphi) = \top$. Further, with $\mathbf{\Phi}$ chosen to be $\mathbf{P}$ and L chosen to be two-valued we completely recover the traditional definition of entailment systems as given in Definition 3.2.1. Indeed, we may make this observation concrete by the following proposition.

Proposition 4.1.2. *Assume a traditional entailment system*

$$\mathcal{E}' = (\mathsf{Sign}, \mathsf{Sen}, \vdash').$$

Then $\mathcal{E}'$ is isomorphic to a generalized entailment system

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

where $\mathsf{L} = \{\top, \bot\}$, $\mathbf{P} = (\mathsf{P}, \subseteq, \eta, \mu)$ is the power set partially ordered monad, and $\vdash$ is a family of two-valued entailment relations such that

$$(\Gamma \vdash \Gamma') = \top \iff (\forall \varphi \in \Gamma')\ \Gamma \vdash' \varphi \tag{49}$$

for all $\Gamma, \Gamma' \in \mathsf{PSen}(\Sigma)$.

Proof. Now, assume that $\mathcal{E}'$ is known, we wish to show that the corresponding $\mathcal{E}$ is a generalized entailment system. For reflexivity, 4.1.1(i), let $\Gamma \in \mathsf{PSen}(\Sigma)$. For each $\varphi \in \Gamma$ we have $\{\varphi\} \vdash' \varphi$ and by monotonicity we have $(\forall \varphi \in \Gamma)\ \Gamma \vdash' \varphi$. From (49) it then follows that $(\Gamma \vdash \Gamma) = \top$, which shows that 4.1.1(i) holds.

For 4.1.1(ii), axiom monotonicity, we observe from 3.2.1(ii) that $\Gamma = \Gamma' \cup \Gamma''$ for some Γ'' and we can equivalently state 3.2.1(ii) as $\Gamma' \vdash' \varphi \implies (\Gamma' \cup \Gamma'') \vdash' \varphi$. Of course it also follows that $\Gamma'' \vdash' \varphi \implies (\Gamma' \cup \Gamma'') \vdash' \varphi$. By extending to sets of consequents and using (49) we find that axiom monotonicity holds.

For 4.1.1(iii) we observe that if both $(\forall \varphi \in \Gamma')\ \Gamma \vdash' \varphi$ and $(\forall \varphi \in \Gamma'')\ \Gamma \vdash' \varphi$ hold then $(\forall \varphi \in \Gamma' \cup \Gamma'')\ \Gamma \vdash' \varphi$ holds. Condition 4.1.1(iii) follows.

For 4.1.1(iv) we find that the traditional entailment system has a similar condition, expressible as $(\forall \varphi \in \Gamma')\ \Gamma \vdash' \varphi$ and $(\forall \varphi \in \Gamma'')\ \Gamma \cup \Gamma' \vdash' \varphi$ imply that $(\forall \varphi \in \Gamma'')\ \Gamma \vdash' \varphi$. Condition 4.1.1(iv) follows directly.

For 4.1.1(v) the result follows from the condition in traditional entailment systems that state, for $\sigma \in \mathrm{Hom}_{\mathtt{Sign}}(\Sigma, \Sigma')$,

$$\Gamma \vdash'_{\Sigma} \varphi \implies \mathsf{PSen}(\sigma)(\Gamma) \vdash'_{\Sigma'} \mathsf{Sen}(\sigma)(\varphi).$$

extending to a set of consequents, Γ', gives

$$(\forall \varphi \in \Gamma')\ \Gamma \vdash'_{\Sigma} \varphi \implies (\forall \varphi \in \Gamma')\ \mathsf{PSen}(\sigma)(\Gamma) \vdash'_{\Sigma'} \mathsf{Sen}(\sigma)(\varphi)$$

translating to the generalized entailment system will then give 4.1.1(v) as desired.

It remains to show that assuming the generalized entailment system $\mathcal{E}$ gives a traditional entailment system. Condition 3.2.1(i) immediately follows from the equivalence between $\{\varphi\} \vdash \{\varphi\}) = \top$ and $\{\varphi\} \vdash' \varphi$. Similarly, for 3.2.1(ii) assume that $(\Gamma \vdash \{\varphi\}) = \top$. Then $((\Gamma \vee \Gamma') \vdash \{\varphi\}) = \top$ holds for all $\Gamma' \in \mathsf{PSen}(\Sigma)$. Mapping to the corresponding traditional entailment system gives that if $\Gamma \vdash' \varphi$ then $(\Gamma \cup \Gamma') \vdash' \varphi$, which has already been shown to be equivalent to 3.2.1(ii).

For 3.2.1(iii) we find that 4.1.1(iv) with $\Gamma' = \{\varphi_i \mid i \in I\}$ and $\Gamma'' = \{\psi\}$ will precisely recover 3.2.1(iii). Finally, 3.2.1(iv) follows from setting $\Gamma' = \{\varphi\}$ in 4.1.1(v). □

As an interesting side-note, similar to the manner in which a relation $R \subseteq A \times B$ equally well may be viewed as a Kleisli morphism $R\colon A \longrightarrow PB$ we may view an L-valued relation such as $\vdash_{\Sigma}$ as a Kleisli morphism $\vdash_{\Sigma}\colon \Delta\mathsf{Sen}(\Sigma) \longrightarrow L\Delta\mathsf{Sen}(\Sigma)$. As a result the conditions of Definition 3.2.1 and Definition 4.1.1 may be reconsidered as identities between morphisms in a Kleisli category. For example, condition 4.1.1(i) becomes $\vdash_{\Sigma}(\Gamma) \geqslant \eta^{\mathbf{L}}_{\Delta\mathsf{Sen}(\Sigma)}(\Gamma)$ and condition 4.1.1(ii) becomes $\vdash_{\Sigma}(\Gamma_1 \vee \Gamma_2) \geqslant \vdash_{\Sigma}(\Gamma_1) \vee \vdash_{\Sigma}(\Gamma_2)$. A similar view may be considered for the following definition of generalized institutions but, while being an interesting viewpoint, we will however not adopt this view here.

Clearly, this move to generalized entailment systems will warrant a corresponding change in the notion of theories. To determine the nature of generalized theories we first must establish a suitable notion of closures. To this end let $\mathcal{E}$ be an entailment system and $\mathfrak{l} \in \mathsf{L}$, then an $\mathfrak{l}$-*closure* over a $\mathcal{E}$-theoremata $\Gamma \in \Delta\mathsf{Sen}(\Sigma)$ will be denoted $\Gamma^{\bullet}_{\mathfrak{l}}$ and contains the sentences provable from Γ with $\mathfrak{l}$ being the minimum degree of certainty. That is, we have

$$\Gamma^{\bullet}_{\mathfrak{l}} = \bigvee_{\Gamma \vdash \Gamma' \geqslant \mathfrak{l}} \Gamma'.$$

Note, we immediately have $\Gamma \vdash \Gamma^{\bullet}_{\mathfrak{l}} \geqslant \mathfrak{l}$ and $\Gamma^{\bullet}_{\bot} = \bigvee \Delta\mathsf{Sen}(\Sigma)$ from consequent invariance (condition 4.1.1(iii)) in the definition of entailment systems.

With the notion of $\mathfrak{l}$-closures we may now introduce generalized theories and their categories. The construction is straightforward and follows the traditional notion closely. Specifically, a theory (Σ, Γ) for an entailment system with sentence functor Sen and signature category $\mathtt{Sign}$ provides the axioms of interest in the form of a theoremata object $\Gamma \in \Delta\mathsf{Sen}(\Sigma)$, as well as the underlying signature $\Sigma \in \mathrm{Ob}(\mathtt{Sign})$.

We may for a generalized entailment system, $\mathcal{E}$, now consider $\mathcal{E}$-theories as objects in a category $\mathsf{Th}(\mathcal{E})$ where a morphism from (Σ_1, Γ_1) to (Σ_2, Γ_2) in $\mathsf{Th}(\mathcal{E})$ consists of a morphism $\sigma\colon \Sigma_1 \longrightarrow \Sigma_2$ in $\mathtt{Sign}$ such that for all $\Gamma' \preceq \Gamma_1$, we have

$$(\Gamma_2 \vdash_{\Sigma_2} \Delta\mathsf{Sen}(\sigma)(\Gamma')) = (\Gamma_1 \vdash_{\Sigma_1} \Gamma') = \top.$$

Further, if

$$\Delta\mathsf{Sen}(\sigma)(\Gamma') \preceq \Gamma_2$$

then the morphism is said to be *axiom preserving* and we define $\widehat{\mathsf{Th}}(\mathcal{E})$ to be the subcategory of $\mathsf{Th}(\mathcal{E})$ containing all objects of $\mathsf{Th}(\mathcal{E})$ but only axiom preserving morphisms. Figure 3 illustrates the notion of axiom and non-axiom preserving theory morphisms, the figure is closely related to Figure 2 but differs in the type of objects forming image and coimage of the morphisms, further the figure in the generalized include fainter circles representing closures with decreasing certainty.

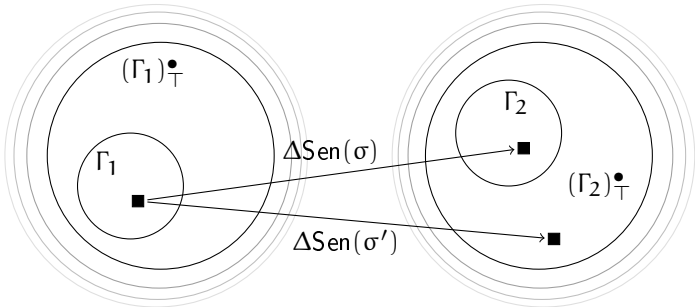


Figure 3: Comparison of axiom-preserving and non axiom-preserving theory morphisms, denoted σ_{Th} and σ'_{Th}, respectively. Here the source square represents some theoremata $\Gamma' \in \Delta\mathsf{Sen}(\Sigma)$ with $\Gamma' \preceq \Gamma_1$

With the definition of generalized entailment systems and theories established, we will immediately consider maps between such entailment systems. These again coincide quite closely to the traditional definition and the necessary extensions are related directly with mapping, e. g., the included partially ordered monads.

Definition 4.1.2. Let

$$\mathcal{E}_1 = (\mathsf{Sign}_1, \mathsf{Sen}_1, \boldsymbol{\Delta}_1, \mathrm{L}_2, \vdash^1)$$

and

$$\mathcal{E}_2 = (\mathsf{Sign}_2, \mathsf{Sen}_2, \boldsymbol{\Delta}_2, \mathrm{L}_2, \vdash^2)$$

be two entailment systems. Then a map from $\mathcal{E}_1$ to $\mathcal{E}_2$ is a structure

$$(\Phi, \tau, \alpha, \mathrm{h}) \colon \mathcal{E}_1 \longrightarrow \mathcal{E}_2$$

where

- $\Phi \colon \widehat{\mathsf{Th}}(\mathcal{E}_1) \longrightarrow \widehat{\mathsf{Th}}(\mathcal{E}_2)$ is a functor from the category of $\mathcal{E}_1$-theories to the category of $\mathcal{E}_2$-theories;
- $\tau \colon \boldsymbol{\Delta}_1 \longrightarrow \boldsymbol{\Delta}_2$ is a partially ordered monad morphism;

- $\alpha\colon \mathsf{Sen}_1 \longrightarrow \mathsf{Sen}_2 \circ \mathcal{S} \circ \Phi \circ \mathcal{T}$ is a natural transformation; and
- $h\colon L_1 \longrightarrow L_2$ is a lattice homomorphism

such that

(i) Φ maps theory signatures with no regard to axioms, that is, $\mathcal{S}\Phi = \mathcal{S}\Phi\mathcal{T}\mathcal{S}$;

(ii) the theories $\Phi(\Sigma, \Gamma)$ and $\Phi(\Sigma, \emptyset)$ are such that

$$(\mathcal{A}\Phi(\Sigma, \Gamma))^{\bullet}_{l} = (\mathcal{A}\Phi(\Sigma, \emptyset) \vee (\tau \star \alpha)_{\Sigma}(\Gamma))^{\bullet}_{l}; \text{ and}$$

(iii) for each $\Sigma \in \mathrm{Ob}(\mathtt{Sign})$,

$$h(\Gamma \vdash^{1}_{\Sigma} \Gamma') \leqslant (\tau \star \alpha)_{\Sigma}(\Gamma) \vee \mathcal{A}'\Phi(\Sigma, \emptyset) \vdash^{2}_{\mathcal{S}'\Phi(\Sigma,\emptyset)} (\tau \star \alpha)_{\Sigma}(\Gamma'). \quad (50)$$

We say that the map (Φ, τ, α) is *conservative* if the inequality of (50) actually is an equality. That is,

$$h(\Gamma \vdash^{1}_{\Sigma} \Gamma') = (\tau \star \alpha)_{\Sigma}(\Gamma) \vee \mathcal{A}\Phi(\Sigma, \emptyset) \vdash^{2}_{\mathcal{S}\Phi(\Sigma,\emptyset)} (\tau \star \alpha)_{\Sigma}(\Gamma')$$

holds.

By defining composition of these entailment morphisms in the obvious component wise manner we arrive at a category $\mathtt{GEnt}$ of generalized entailment systems.

Turning to the semantic side of logic we may now consider generalized institutions. Given the construction of generalized entailment system, the corresponding generalized institution follows the construction closely and offers few surprises.

Definition 4.1.3. A generalized institution

$$\mathcal{I} = (\mathtt{Sign}, \mathsf{Sen}, \mathsf{Mod}, \boldsymbol{\Delta}, L, \models)$$

is a structure where

- $\mathtt{Sign}$ is a category of signatures;
- Sen is a functor $\mathsf{Sen}\colon \mathtt{Sign} \longrightarrow \mathtt{Set}$ taking signatures to sentences,

- $\mathsf{Mod}\colon \mathtt{Sign} \longrightarrow \mathtt{Cat}^{\mathrm{op}}$ is a functor with $\mathsf{Mod}(\Sigma)$ representing the category of Σ-*models*;
- $\boldsymbol{\Delta} = (\Delta, \preceq, \eta, \mu)$ is a partially ordered monad over $\mathtt{Set}$, typically called the *theoremata* functor;
- L is a completely distributive lattice; and
- $\models$ is a family of L-valued relations consisting of

  $$\models_\Sigma\colon \mathrm{Ob}(\mathsf{Mod}(\Sigma)) \times \Delta\mathsf{Sen}(\Sigma) \longrightarrow \mathrm{L}$$

  for each signature $\Sigma \in \mathrm{Ob}(\mathtt{Sign})$ where $\models_\Sigma$ is called a Σ-*satisfaction relation*.

The $\models_\Sigma$ relations must fulfill the *satisfaction condition* that states that for all signature morphisms $\sigma \in \mathrm{Hom}_{\mathtt{Sign}}(\Sigma, \Sigma')$, models $M \in \mathrm{Ob}(\mathsf{Mod}(\Sigma))$ and theoremata $\Gamma \in \Delta\mathsf{Sen}(\Sigma)$, $\models_\Sigma$ must be such that

$$(\mathsf{Mod}(\sigma)(M) \models_\Sigma \Gamma) = (M \models_{\Sigma'} \Delta\mathsf{Sen}(\sigma)(\Gamma)). \tag{51}$$

Again, by the same argument as for entailment systems we choose to interpret $M \models_\Sigma \varphi$ to mean as $M \models_\Sigma \eta_{\mathsf{Sen}(\Sigma)}(\varphi)$. Further, as was the case for traditional institutions, we often wish to view the satisfaction relation in the context of the theoremata satisfied by these models. The construction is similar to the one in Definition 3.1.2 but since $\models$ now is L-valued we must, however, introduce a limit value $l \in \mathrm{L}$ indicating the least acceptable degree of truth. Then for some model $M \in \mathsf{Mod}(\Sigma)$ we may determine a theoremata $\Gamma \in \Delta\mathsf{Sen}(\Sigma)$ by

$$\Gamma = \bigvee\{\Gamma' \in \Delta\mathsf{Sen}(\Sigma) \mid M \models_\Sigma \Gamma' \geqslant l\}.$$

From this type of representation, we are able to construct a full subcategory of $\mathsf{Mod}(\Sigma)$, denoted $\mathsf{Mod}(\Sigma, \Gamma, l)$. This subcategory has as objects all models that satisfy the theoremata Γ to at least degree $l \in \mathrm{L}$. This in turn now allow us to define the alternate family of satisfaction relations.

Definition 4.1.4. Let $\mathcal{I} = (\mathtt{Sign}, \mathsf{Sen}, \mathsf{Mod}, \boldsymbol{\Delta}, \mathbf{L}, \models)$ be an arbitrary generalized institution and $\widetilde{\models}$ be a family of $\mathbf{L}$-valued relations

$$\widetilde{\models}_{\Sigma} : \Delta\mathsf{Sen}(\Sigma) \times \Delta\mathsf{Sen}(\Sigma) \longrightarrow \mathbf{L}$$

for each signature $\Sigma \in \mathrm{Ob}(\mathtt{Sign})$. Then each $\widetilde{\models}_{\Sigma}$ is also said to be a satisfaction relation if the condition

$$(\Gamma \mathrel{\widetilde{\models}_{\Sigma}} \Gamma') = (M \models_{\Sigma} \Gamma')$$

holds for all $\mathfrak{l} \in \mathbf{L}$, $\Gamma \in \Delta\mathsf{Sen}(\Sigma)$ and $M \in \mathsf{Mod}(\Sigma, \Gamma, \top)$. Again, by abuse of notation we will simply write $\models$ even when $\widetilde{\models}$ is intended.

Using this alternate relation we may define a theory from an institution defining for a theoremata Γ its $\mathfrak{l}$-closure by

$$\Gamma^{\bullet}_{\mathfrak{l}} = \bigvee_{\Gamma \models \Gamma' \geqslant \mathfrak{l}} \Gamma'$$

and further define for a generalized institution $\mathcal{I}$ the theories and their categories $\mathsf{Th}(\mathcal{I})$ and $\widehat{\mathsf{Th}}(\mathcal{I})$ using the same argumentation as in the case of entailment systems.

Similarly to the case of traditional and generalized entailment systems, we can make the observation that the traditional institution is easily recovered by letting $\boldsymbol{\Delta}$ be the powerset monad and $\mathbf{L}$ be a two point element set, e. g., $\mathbf{L} = \{\top, \bot\}$.

Proposition 4.1.3. *Assume a traditional institution in the sense of Definition 3.1.1, say*

$$\mathcal{I}' = (\mathtt{Sign}, \mathsf{Sen}, \mathsf{Mod}, \models').$$

Then $\mathcal{I}'$ is isomorphic to a generalized institution

$$\mathcal{I} = (\mathtt{Sign}, \mathsf{Sen}, \mathsf{Mod}, \boldsymbol{\Delta}, \mathbf{L}, \models)$$

where $\boldsymbol{\Delta} = \mathbf{P}$, $\mathbf{L}$ being a two valued lattice, e. g., $\mathbf{L} = \{\top, \bot\}$, and $\models$ is a family of two-valued relations such that

$$(M \models \Gamma) = \top \iff (\forall \varphi \in \Gamma)\ M \models' \varphi \tag{52}$$

for all $M \in \mathrm{Ob}(\mathsf{Mod}(\Sigma))$ and $\Gamma \in \mathsf{PSen}(\Sigma)$.

Proof. It suffices to show that the corresponding generalized institution for a given traditional institution uphold the satisfaction condition and vice versa. From the satisfaction condition of $\models'$ we know that

$$\mathsf{Mod}(\sigma)(M) \models'_{\Sigma} \varphi \iff M \models'_{\Sigma'} \mathsf{Sen}(\sigma)(\varphi)$$

for all $\varphi \in \mathsf{Sen}(\Sigma)$. Then it follows immediately that for some $\Gamma \in \mathsf{PSen}(\Sigma)$, we have

$$(\forall \varphi \in \Gamma)\ \mathsf{Mod}(\sigma)(M) \models'_{\Sigma} \varphi \iff (\forall \varphi \in \Gamma)\ M \models'_{\Sigma'} \mathsf{Sen}(\sigma)(\varphi).$$

By (52) and the definition of P, this becomes

$$(\mathsf{Mod}(\sigma)(M) \models_{\Sigma} \Gamma) = \top \iff (M \models_{\Sigma'} \mathsf{PSen}(\sigma)(\Gamma)) = \top$$

which is equivalent to

$$(\mathsf{Mod}(\sigma)(M) \models_{\Sigma} \Gamma) = (M \models_{\Sigma'} \mathsf{PSen}(\sigma)(\Gamma)).$$

This is precisely the satisfaction condition of the corresponding generalized institution and the isomorphism sought for has thus been shown. Showing that the generalized institution give rise to a traditional institution follows the opposite argumentation. □

Returning to the notion of maps we have the following construction for maps of institutions. Note, defining composition component-wise again gives a category of institutions, `GInst`.

Definition 4.1.5. Let

$$\mathcal{I}_1 = (\mathtt{Sign}_1, \mathsf{Sen}_1, \mathsf{Mod}_1, \boldsymbol{\Delta}_1, \mathsf{L}_1, \models^1) \quad \text{and}$$
$$\mathcal{I}_2 = (\mathtt{Sign}_2, \mathsf{Sen}_2, \mathsf{Mod}_2, \boldsymbol{\Delta}_2, \mathsf{L}_2, \models^2)$$

be two institutions. Then a map from $\mathcal{I}_1$ to $\mathcal{I}_2$ is a structure

$$(\Phi, \tau, \alpha, \beta, \mathfrak{h}) \colon \mathcal{I}_1 \longrightarrow \mathcal{I}_2$$

where

- $\Phi \colon \widehat{\mathsf{Th}}(\mathcal{I}_1) \longrightarrow \widehat{\mathsf{Th}}(\mathcal{I}_2)$ is a functor;

- $\tau\colon \boldsymbol{\Delta}_1 \longrightarrow \boldsymbol{\Delta}_2$ is a partially ordered monad morphism;
- $\alpha\colon \mathsf{Sen}_1 \longrightarrow \mathsf{Sen}_2 \circ \mathcal{S} \circ \Phi \circ \mathcal{T}$ is a natural transformation;
- $\beta\colon \mathsf{Mod}_2 \circ \mathcal{S} \circ \Phi \circ \mathcal{T} \longrightarrow \mathsf{Mod}_1$ is a natural transformation; and
- $h\colon L_1 \longrightarrow L_2$ is a lattice homomorphism.

These are subject to the conditions

(i) $\mathcal{S}\Phi = \mathcal{S}\Phi\mathcal{T}\mathcal{S}$;

(ii) $(\mathcal{A}\Phi(\Sigma,\Gamma))^{\bullet}_{l} = (\mathcal{A}\Phi(\Sigma,\emptyset) \vee (\tau \star \alpha)_{\Sigma}(\Gamma))^{\bullet}_{l}$; and

(iii) for each $\Sigma \in \mathrm{Ob}(\mathsf{Sign})$, $\Gamma \in \Delta\mathsf{Sen}(\Sigma)$, and $M' \in \mathsf{Mod}_2(\Sigma')$ where $\Sigma' = \mathcal{S}\Phi\mathcal{T}(\Sigma)$, we have

$$M' \models^{2}_{\Sigma} (\tau \star \alpha)_{\Sigma}(\Gamma) = h(\beta_{\Sigma}(M') \models^{1}_{\Sigma} \Gamma).$$

Note, much like in the traditional case we may immediately observe that the two first conditions of this definition are the same as the two first conditions of maps entailment system.

Now, as indicated in the previous chapter, it is typically of great interest to combine the notion of an entailment relation with the notion of a satisfaction relation. These relations are interrelated using a soundness condition and often also a completeness condition. This combination is called a *logic* and we extend its traditional definition, given in Definition 3.3.1, into the notion of generalized logic as given below.

Definition 4.1.6. A generalized logic $\mathcal{L}$ is a structure

$$\mathcal{L} = (\mathsf{Sign}, \mathsf{Sen}, \mathsf{Mod}, \boldsymbol{\Delta}, L, \vdash, \models)$$

such that

- $(\mathsf{Sign}, \mathsf{Sen}, \boldsymbol{\Delta}, L, \vdash)$ is an generalized entailment system;
- $(\mathsf{Sign}, \mathsf{Sen}, \mathsf{Mod}, \boldsymbol{\Delta}, L, \models)$ is a generalized institution; and
- the condition

  $$(\Gamma \vdash_{\Sigma} \Gamma') \leqslant (\Gamma \models_{\Sigma} \Gamma')$$

  holds for all $\Sigma \in \mathrm{Ob}(\mathsf{Sign})$, $\Gamma, \Gamma' \in \Delta\mathsf{Sen}(\Sigma)$.

The last condition is commonly called the *soundness condition*. If the logic also fulfill the stronger condition

$$(\Gamma \vdash_\Sigma \Gamma') = (\Gamma \models_\Sigma \Gamma')$$

then it is said to be *complete*.

Again, for this notion of logic we introduce maps between logics in a way very much similar to the tradition definition. With composition defined component-wise and identity mappings being the obvious one we arrive at a category `GLog` of generalized logics and their mappings.

Definition 4.1.7. Let

$$\mathcal{L}_1 = (\mathtt{Sign}_1, \mathtt{Sen}_1, \mathtt{Mod}_1, \boldsymbol{\Delta}_1, L_1, \vdash^1, \models^1) \text{ and}$$
$$\mathcal{L}_2 = (\mathtt{Sign}_2, \mathtt{Sen}_2, \mathtt{Mod}_2, \boldsymbol{\Delta}_2, L_2, \vdash^2, \models^2)$$

be two logics, then a *map of logics* is a structure

$$(\Phi, \tau, \alpha, \beta, h) \colon \mathcal{L}_1 \longrightarrow \mathcal{L}_2$$

such that

$$(\Phi, \tau, \alpha, \beta, h) \colon (\mathtt{Sign}_1, \mathtt{Sen}_1, \mathtt{Mod}_1, \boldsymbol{\Delta}_1, L_1, \models^1) \longrightarrow (\mathtt{Sign}_2, \mathtt{Sen}_2, \mathtt{Mod}_2, \boldsymbol{\Delta}_2, L_2, \models^2)$$

is a map of the underlying institutions, and

$$(\Phi, \tau, \alpha, h) \colon (\mathtt{Sign}_1, \mathtt{Sen}_1, \boldsymbol{\Delta}_1, L_1, \vdash^1) \longrightarrow (\mathtt{Sign}_2, \mathtt{Sen}_2, \boldsymbol{\Delta}_2, L_2, \vdash^2)$$

is a map of the underlying entailment systems.

As in the traditional case, we can associate to an entailment system a proof calculi and as previously mentioned we make these constructions separate because a single logic may have several possible proof calculi. Clearly, the addition of non-binary notions of truth complicates the issue of proof calculi to a certain extent and indeed, this may often make it intractable to establish complete non-classical logics. Fortunately, soundness is often

sufficient and for such cases we may in practical cases look for proofs showing satisfaction to at least some degree $\mathfrak{l} \in \mathbf{L}$ rather than to exactly $\mathfrak{l}$.

Nevertheless, proof calculi in their general forms are rather similar to the traditional case in that it contains similar components and we merely adopt them to the L-valued nature of the $\vdash$ relation.

Definition 4.1.8. A generalized proof calculus is a structure $\mathcal{P} = (\mathcal{E}, \mathsf{Pt}, \mathsf{Pr}, \pi)$ such that

(i) $\mathcal{E} = (\mathtt{Sign}, \mathsf{Sen}, \Delta, \mathbf{L}, \vdash)$ is an entailment system;

(ii) $\mathsf{Pt}\colon \widehat{\mathsf{Th}}(\mathcal{E}) \longrightarrow \mathtt{Struct}$ is a functor taking theories to corresponding proof-theoretic structures;

(iii) for each $\mathfrak{l} \in \mathbf{L}$, $\mathsf{Pr}_\mathfrak{l}\colon \mathtt{Struct} \longrightarrow \mathtt{Set}$ is a functor such that for $\mathbf{T} \in \mathrm{Ob}(\widehat{\mathsf{Th}}(\mathcal{E}))$ we have that $(\mathsf{Pr}_\mathfrak{l} \circ \mathsf{Pt})(\mathbf{T})$ represents the *set of* $\mathfrak{l}$*-level proofs for* $\mathbf{T}$; and

(iv) for each $\mathfrak{l} \in \mathbf{L}$, $\pi^\mathfrak{l}\colon \mathsf{Pr}_\mathfrak{l} \circ \mathsf{Pt} \longrightarrow \mathsf{Sen} \circ \mathcal{S}$ is a natural transformation such that the image of the morphism

$$\pi^\mathfrak{l}_\mathbf{T}\colon (\mathsf{Pr}_\mathfrak{l} \circ \mathsf{Pt})(\mathbf{T}) \longrightarrow (\Delta \circ \mathsf{Sen} \circ \mathcal{S})(\mathbf{T}),$$

for some theory $\mathbf{T} = (\Sigma, \Gamma)$, is the theoremata $\Gamma^\bullet_\mathfrak{l}$.

Note that `Struct` at this point remains an abstract category of proof structures. We will, however, make this category more concrete when moving to substitution logics. First, we must consider maps of proof calculi, however. These again are a conservative generalization of the conventional definition. In particular, we must include a lattice homomorphism to associate truth degrees in the source proof calculi to the truth degrees of the target calculi. Clearly, this also means that the families Pr and π must be associated in a way consistent with the lattice homomorphism. The following definition makes these considerations concrete.

Definition 4.1.9. Let

$$\mathcal{P}_1 = (\mathcal{E}_1, \mathsf{Pt}_1, \mathsf{Pr}_1, \pi^1) \text{ and}$$
$$\mathcal{P}_2 = (\mathcal{E}_2, \mathsf{Pt}_2, \mathsf{Pr}_2, \pi^2)$$

be two generalized proof calculi. Then a *map of proof calculi*

$$(\Phi, \tau, \alpha, h, \gamma) \colon \mathcal{P}_1 \longrightarrow \mathcal{P}_2$$

is such that

$$(\Phi, \tau, \alpha, h) \colon \mathcal{E}_1 \longrightarrow \mathcal{E}_2$$

maps the associated entailment system of $\mathcal{P}_1$ to the one of $\mathcal{P}_2$. Further,

$$\gamma^l \colon (\mathsf{Pr}_1)_l \circ \mathsf{Pt}_1 \longrightarrow (\mathsf{Pr}_2)_{h(l)} \circ \mathsf{Pt}_2 \circ \Phi$$

is a natural transformation such that

$$\alpha\mathcal{S} \circ (\pi^1)^l = (\pi^2)^{h(l)}\Phi \circ \gamma^l. \tag{53}$$

Finally, when a generalized logic is given a specific generalized proof calculus the combined result is known as a generalized logical system. The definition is straight-forward.

Definition 4.1.10. A generalized logical system is an structure

$$(\mathsf{Sign}, \mathsf{Sen}, \mathsf{Mod}, \boldsymbol{\Delta}, L, \vdash, \models, \mathsf{Pt}, \mathsf{Pr}, \pi)$$

where

(i) $(\mathsf{Sign}, \mathsf{Sen}, \mathsf{Mod}, \boldsymbol{\Delta}, L, \vdash, \models)$ is a generalized logic and

(ii) $(\mathsf{Sign}, \mathsf{Sen}, \boldsymbol{\Delta}, L, \vdash, \mathsf{Pt}, \mathsf{Pr}, \pi)$ is a generalized proof calculus.

Clearly, we can now construct the category, `Glossy's`, of generalized logical systems by combining maps of logics and maps of proof calculi.

Definition 4.1.11. Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be two generalized logical systems, then a map from one to the other is a structure

$$(\Phi, \tau, \alpha, \beta, h, \gamma) \colon \mathcal{S}_1 \longrightarrow \mathcal{S}_2$$

such that

$$(\Phi, \alpha, \beta, \gamma \colon \mathrm{logic}(\mathcal{S}_1) \longrightarrow \mathrm{logic}(\mathcal{S}'_2)$$

maps the logic component of $\mathcal{S}_1$ to its counterpart in $\mathcal{S}_2$ and

$$(\Phi, \tau, \alpha, h, \gamma)\colon \mathrm{pcalc}(\mathcal{S}_1) \longrightarrow \mathrm{pcalc}(\mathcal{S}_2)$$

similarly maps the proof calculus of $\mathcal{S}_1$ to the proof calculus of $\mathcal{S}_2$.

This concludes the generalized constructions introduced over Meseguer's general logics. We've added generalized notions for the central concepts of general logics. While additional components exist in general logics, most importantly the proof calculi, remain as future work we have shown a logical framework that give an approachable means of describing and reasoning about non-classical notions of logic.

To end the section we will consider a simple example using the NATADD signature – however, for brevity, we let the succ operation be called S. Note, we will consider the classical case with crisp power sets as compared to the non-classical case with many-valued power sets. We will for these examples informally define a suitable proof calculi. In subsequent papers we will provide formal treatments and extensions of proof calculi, and also include more examples such as those involving the partially ordered fuzzy filter monad.

For the crisp power set we have the standard definition of addition

$$\begin{aligned}\Gamma = \{&a + 0 = a,\\ &a + S(b) = S(a + b)\}\end{aligned}$$

followed by the traditional proof calculi for equational logic as previously given.

In the case of the many-valued power set partially ordered monad we have axioms of the form $\Gamma\colon \mathsf{Sen}(\Sigma) \longrightarrow L$ with $X = \{a, b\}$ being our set of variables. For this example, L is the unit interval and it is also used to indicate the truth values of the $\vdash$ relation. We shall define Γ as per

$$\begin{aligned}\Gamma(a + 0 = a) &= 1.0,\\ \Gamma(a + S(b) = S(a + b)) &= 0.95\end{aligned}$$

and $\Gamma(x) = 0$ for all other $x \in \mathsf{Sen}(\Sigma)$. Intuitively we associate a degree of confidence to the two axioms – we are certain of the first axiom but slightly uncertain of the second. Using Γ, we now attempt to find the value of

$$\Gamma \vdash (\mathsf{S}(\mathsf{S}(0)) + 0) + \mathsf{S}(0) = \mathsf{S}(\mathsf{S}(0)) + (0 + \mathsf{S}(0)) : 0.95 \qquad (54)$$

where $a : x$ indicate that the sentence a has x as degree of membership.

To do this we first introduce the generalized deduction rules used in this example. Note, we will here in the case of substitution introduce slight fuzziness into the rule itself, indicated by the scaling factor 0.99 in SUBST. Thus for $t, u, v \in \mathsf{T}_\Sigma X$, $x, y \in L$, and substitution $\sigma : X \longrightarrow \mathsf{T}_\Sigma Y$ we have

$$\frac{}{t = t : 1.0}\ \text{REFL}$$

$$\frac{t = u : x}{u = t : x}\ \text{COMM}$$

$$\frac{t = u : x \qquad u = v : y}{t = v : x \cdot y}\ \text{TRANS}$$

$$\frac{t = u : x}{t\sigma = u\sigma : 0.99 \cdot x}\ \text{SUBST}$$

$$\frac{t_1 = t_1' : x_1 \quad \cdots \quad t_n = t_n' : x_n}{f(t_1, \ldots, t_n) = f(t_1', \ldots, t_n') : x_1 \cdot \cdots \cdot x_n}\ \text{CONG}$$

as deductive rules with the rule names given in the right column. Note the use of x and y to refer to degree of membership of the source sentence, these values are then used, together with the rule fuzziness, to calculate the target sentence membership. Of

course, we may also make use of our axioms, e. g., $a + 0 = a : 1.0$. Using these rules we can now construct the derivation sequence

$$\begin{aligned}
(\mathtt{SS0} + \mathtt{S0}) + \mathtt{S0} &= \mathtt{S}((\mathtt{SS0} + \mathtt{S0}) + \mathtt{0}) : (0.99 \cdot 0.95 = 0.94) \\
&= \mathtt{S}(\mathtt{SS0} + \mathtt{S0}) : (0.99 \cdot 0.94 = 0.93) \\
&= \mathtt{SS}(\mathtt{SS0} + \mathtt{0}) : (0.99 \cdot 0.95 \cdot 0.93 = 0.87) \\
&= \mathtt{SSSS0} : (0.99 \cdot 0.87 = 0.86) \\
&= \mathtt{SS}(\mathtt{SS0} + \mathtt{0}) : (0.99 \cdot 0.86 = 0.85) \\
&= \mathtt{S}(\mathtt{SS0} + \mathtt{S0}) : (0.99 \cdot 0.95 \cdot 0.85 = 0.80) \\
&= \mathtt{SS0} + \mathtt{SS0} : (0.99 \cdot 0.95 \cdot 0.80 = 0.75) \\
&= \mathtt{SS0} + \mathtt{S}(\mathtt{S0} + \mathtt{0}) : (0.99 \cdot 0.75 = 0.74) \\
&= \mathtt{SS0} + (\mathtt{S0} + \mathtt{S0}) : (0.99 \cdot 0.95 \cdot 0.74 = 0.70)
\end{aligned}$$

Note, in the derivation sequence we find 0.99 appearing whenever a substitution occurs and 0.95 appears whenever we make use of the second axiom. The sequence shows that the desired equation is derivable using our deductive rules, but the degree of membership is less than desired, only 0.70 while we expected 0.95. We let this discrepancy be reflected in the value of our $\vdash$ relation, that is, the further from our expected membership the lower the truth of our entailment. This notion can, for example, be captured by letting

$$\Gamma \vdash \varphi : x = 1.0 - |x - y|$$

where y is the value found using our deductive process. In (54), this would mean that the value of the entailment relation is

$$1.0 - |0.95 - 0.70| = 0.75.$$

4.2 SUBSTITUTION LOGIC

Clearly, the generalized notions of logic of the last section are in many cases too general. In particular, many logics share an underlying structure based on more or less traditional terms. We may therefore beneficially provide a logic framework based on generalized general logic that explicitly states that sentences are

constructed using a term monad as base. Note, here the term monad does not necessarily coincide with the term monads of Section 2.7, although they in practice often do. To this end, we will in this section briefly outline a useful specialization of generalized general logic that is aimed for this common case of logics based on more standard notions of signatures and terms and whose inference mechanisms rely on substitutions (and unification) in a general sense. Consequently, we will call this new framework *substitution logic*.

The cornerstone of substitution logics is a functor taking signatures to term monads, i. e., $\Theta\colon \mathtt{Sign} \longrightarrow \mathtt{Mnd}[\mathtt{Var}]$ with $\mathtt{Mnd}[\mathtt{Var}]$ being the category of monads over some category $\mathtt{Var}$ of variables. Given a signature Σ, let $\Upsilon(\Theta(\Sigma))$ be the underlying functor of the $\Theta(\Sigma)$ monad. We call Θ a *term monad constructor* and $\Theta(\Sigma)$ the *term monad over the signature* Σ.

We may now consider the $\mathtt{Sen}$ functor in substitution logic as a *sentence functor constructor* in the sense that it takes a term monad to a *sentence functor* over that monad. More specifically – recalling the functor category $[\mathtt{Var}, \mathtt{Set}]$, having as objects functors from $\mathtt{Var}$ to $\mathtt{Set}$ – we have

$$\mathtt{Sen}\colon \mathtt{Mnd}[\mathtt{Var}] \longrightarrow [\mathtt{Var}, \mathtt{Set}],$$

that is, for any monad $\mathbf{F} \in \mathrm{Ob}(\mathtt{Mnd}[\mathtt{Var}])$ we have a functor

$$\mathtt{Sen}(\mathbf{F})\colon \mathtt{Var} \longrightarrow \mathtt{Set}$$

taking some object of variables to sentences over that object. We frequently let $\mathtt{Sen}^{\mathbf{F}}$ denote the functor $\mathtt{Sen}(\mathbf{F})$. Thus, for a specific $\mathtt{Sign}$, a selected signature $\Sigma \in \mathrm{Ob}(\mathtt{Sign})$, and a correspondingly chosen term monad constructor $\Theta\colon \mathtt{Sign} \longrightarrow \mathtt{Mnd}[\mathtt{Var}]$, we have $\mathtt{Sen}^{\Theta(\Sigma)}$ as the *sentence functor over the signature* Σ. Finally, for an $X \in \mathrm{Ob}(\mathtt{Var})$ we then have $\mathtt{Sen}^{\Theta(\Sigma)}(X) \in \mathrm{Ob}(\mathtt{Set})$ as the set of 'legal' sentences in our substitution logic. Much like in the generalized general logic above, for a partially ordered monad $\Delta\colon \mathtt{Set} \longrightarrow \mathtt{Set}$, we have elements $\Gamma \in \Delta\mathtt{Sen}^{\Theta(\Sigma)}(X)$ as *theoremata*.

Note, given a term monad functor Θ and sentence functor constructor $\mathtt{Sen}$ as well as fixing a variables objects $X \in \mathtt{Var}$ immediately lets us establish a 'traditional' sentence functor in the gen-

eral logics sense. Let $\mathsf{TradSen}\colon \mathtt{Sign} \longrightarrow \mathtt{Set}$ be this functor, then define

$$\mathsf{TradSen}(\Sigma) = \mathsf{Sen}^{\Theta(\Sigma)}(X)$$

which is the basis for saying that substitution logics is a specialization of generalized general logics. The benefit of this split is deeply entwined firstly with the notion of $\mathbf{F}$-algebras providing a basis for a semantic interpretation in the logic and secondly with exposing the underlying terms and variables necessary to consider variable substitution in the Kleisli sense and, by extension, use the idea that the traditional unification algorithm correspond to finding a coequalizer in a Kleisli category associated with the underlying term monad [67].

The construction then proceeds with entailment system, institution, etc. as previously. We will here not be specific and instead focus only on the differences from the previously given constructions from generalized general logic. Thus, the construction of *substitution entailment system* is precisely the generalized entailment system having its sentence functor split as per the discussion above. A consequence of this split is that the variable objects now are explicit and must be included in the indexing of our $\vdash$ relations. That is, we now have (Σ, X)-entailment in the form

$$\vdash_{(\Sigma,X)}\colon \Delta\mathsf{Sen}^{\Theta(\Sigma)}(X) \times \Delta\mathsf{Sen}^{\Theta(\Sigma)}(X) \longrightarrow \mathbf{L}$$

for each signature $\Sigma \in \mathrm{Ob}(\mathtt{Sign})$ and variable object $X \in \mathrm{Ob}(\mathtt{Var})$. We must make a corresponding change in our notion of theories, i. e., the variable object must be included. Thus, in substitution logics, theory provides the axioms of interest as well as the signature and variable objects of discourse. Let $\mathcal{E}$ be a substitution entailment system, then a theory in $\mathcal{E}$ is a triple (Σ, X, Γ) where

1. $\Sigma \in \mathrm{Ob}(\mathtt{Sign})$ is a signature;
2. $X \in \mathrm{Ob}(\mathtt{Var})$ is a variable object; and
3. $\Gamma \in \Delta\mathsf{Sen}^{\Theta(\Sigma)}(X)$ are the axioms.

As in the previous construction, this notion of theory also extend to the substitution institution case.

For a semantic treatment of substitution logic recall that all monads $\mathbf{F}$ over some category $\mathtt{Var}$ give rise to a corresponding Eilenberg–Moore category $\mathtt{Var}^{\mathbf{F}}$ having as objects all $\mathbf{F}$-algebras. Now, let $\mathtt{Alg}$ be the category of all Eilenberg–Moore categories, having functors as morphisms. In particular, we are interesting in the subcategory $\mathtt{Alg}^*$ of $\mathtt{Alg}$ having as morphisms functors

$$\mathsf{T}_\tau \colon \mathtt{Var}^{\mathsf{T}_1} \longrightarrow \mathtt{Var}^{\mathsf{T}_2}$$

generated by monad morphisms $\tau \colon \mathbf{T}_1 \longrightarrow \mathbf{T}_2$. We will further use the subcategory $\mathtt{Alg}^{(*,\Theta)}$ restricting $\mathtt{Alg}^*$ to those monads that are provided by Θ, i. e., to the monads $\Theta(\Sigma)$ for all signatures $\Sigma \in \mathrm{Ob}(\mathtt{Sign})$.

It is now possible to introduce term valuation in accordance with the technique used in Section 2.7.3, i. e., for an $A \in \mathrm{Ob}(\mathtt{Var})$, let $\mathfrak{A} = (A, \mathfrak{h})$ be an $\Theta(\Sigma)$-algebra. Then a variable assignment with respect to X and $\mathfrak{A}$, i. e., an assignment for variables in X to values in $\mathfrak{A}$, is a morphism $\mathfrak{v}^{X,\mathfrak{A}} \colon X \longrightarrow A$, where the term $\mathfrak{A}$-evaluation is

$$\mathfrak{h} \circ \widehat{\sigma}_A \circ \Upsilon(\Theta(\Sigma))\mathfrak{v}^{X,\mathfrak{A}} \colon \Upsilon(\Theta(\Sigma))X \longrightarrow A.$$

For a term $t \in \Upsilon(\Theta(\Sigma))X$, note how $\Upsilon(\Theta(\Sigma))\mathfrak{v}^{X,\mathfrak{A}}(t)$ is the syntactic term in the appearance as its semantic counterpart, but still using its syntactical operators, and $\widehat{\sigma}_A \circ \Upsilon(\Theta(\Sigma))\mathfrak{v}^{X,\mathfrak{A}}(t)$ is correspondingly the term in its semantic appearance using its semantic operators. Finally, $\mathfrak{h} \circ \widehat{\sigma}_A \circ \Upsilon(\Theta(\Sigma))\mathfrak{v}^{X,\mathfrak{A}}(t)$ represents 'the value of t in A', as the computed value given the semantics of the operators.

We may now consider substitution institutions as having as base a functor $\mathsf{Mod} \colon \mathtt{Sign} \longrightarrow (\mathtt{Alg}^{(*,\Theta)})^{\mathrm{op}}$ such that

$$\mathsf{Mod}(\Sigma) = \mathtt{Var}^{\Theta(\Sigma)},$$

thus representing the category of Σ-*models*. The $\models$ relation change accordingly to become

$$\models_{(\Sigma,X)} \colon \mathrm{Ob}(\mathsf{Mod}(\Sigma)) \times \Delta\mathsf{Sen}^{\Theta(\Sigma)}(X) \longrightarrow \mathsf{L}$$

and it must fulfill a slightly adapted satisfaction condition stating that for all $\mathtt{Sign}$-morphisms $\sigma \colon \Sigma \longrightarrow \Sigma'$, algebras $\mathfrak{A}' = (A', \mathfrak{h}') \in$

$\mathsf{Ob}(\mathsf{Mod}(\Sigma'))$ and $\mathfrak{A} = (A, \mathfrak{h}) = \mathsf{Mod}(\sigma)(\mathfrak{A}')$, further with $\mathtt{Var}$-morphisms $\mathfrak{v} : X \longrightarrow A$ and $\mathfrak{v}' : X' \longrightarrow A'$ as respective variable assignments, and finally sentences $\Gamma \in \Delta\mathsf{Sen}^{\Theta(\Sigma)}(X)$, for $\models_{(\Sigma,X)}$ we must have that

$$(\mathfrak{A} \models_{(\Sigma,X)} \Gamma) = (\mathfrak{A}' \models_{(\Sigma',X')} \Delta\mathsf{Sen}(\Theta\sigma)(f)(\Gamma)),$$

where $\sigma\colon \Sigma \longrightarrow \Sigma'$ is a $\mathtt{Sign}$-morphism, $f\colon X \longrightarrow X'$ is a $\mathtt{Var}$-morphism, and

$$\mathfrak{v} = \mathsf{Mod}(\sigma) \circ \mathfrak{v}' \circ f. \tag{55}$$

Intuitively, this construction states that the interpretation of a logic's semantics, i. e., the satisfaction relation of the logic, consists of choosing an $\mathbf{F}$-algebra that is then used to evaluate the underlying terms. These evaluated values may then be used by $\models$ to determine the validity of the sentence. In the case that sentences are equalities we may simple compare the term valuations. The idea with this sort of arrangement is clearly the notion of "separation of concerns" and to expose the compositionality of the underlying structures.

We could now go on to construct logics, proof calculi, and so on as per the generalized general logics but with sentence functors substituted in line with the above constructions. More elaborate constructions are possible. We could, e. g., go further in the case of proof calculi to exploit the sorted categories and term monads to capture the nature of "proof terms" in a deep sense, including non-classical cases. Very briefly, let (Σ, X, Γ) be some theory in substitution logic. We may then construct a new many-sorted signature $\mathrm{PRF} = (\Delta\mathsf{Sen}^{\Theta(\Sigma)}(X), \Omega)$ such that

$$\begin{aligned}\Omega = \{&\mathfrak{m}_1 : \ \to \Gamma\} \cup \\ &\{\mathfrak{m}_2 : A \to B \mid A, B \in \Delta\mathsf{Sen}^{\Theta(\Sigma)}(X), B \preceq A\} \cup \\ &\{\mathfrak{m}_3 : A \times B \to A \vee B \mid A, B \in \Delta\mathsf{Sen}^{\Theta(\Sigma)}(X), A \wedge B \succ \bot\} \\ &\vdots\end{aligned}$$

The three given operators then correspond to inference rules typically present in all substitution logics, additional custom rules

may of course be given to correspond with the entailment system. Now a proof of a theoremata $A \in \Delta\mathsf{Sen}^{\Theta(\Sigma)}(X)$ corresponds exactly to a term $t \in \mathsf{T}_{\mathrm{PRF},A}(\emptyset)$.

If our logic is two-valued then the existence of such a proof term is sufficient to determine the truth of the sentence. If, however, our logic of interest is many-valued, then we must now proceed to attach theoremata with values in L, the lattice of truth values. This is enabled by attaching uncertainties to the inference rules, that is, we establish the PRF signature over, e. g., $\mathtt{Set}(\mathfrak{L})$. The degree of membership of the proof term then corresponds to the truth degree of the sentence under the given theory. The full and formal construction of this form of substitution proof calculi remains as future work.

5 APPLICATIONS IN COMPUTER SCIENCE AND ENGINEERING

We will in these following sections explore some of the consequences and applications of, primarily, our term monad construction and, more superficially, the generalized general logic framework. These constructions span the fields of type theory and lambda calculus to description logic and social choice. While quite disparate fields of study we see that their underlying notions of terms have much in common, in particular their use of several levels of signatures and terms, i. e., a signature of one level has as sorts the ground terms of the preceding level.

5.1 TERMS IN TYPE THEORY

Similar to the informal definition of the set of terms, we typically also have the set of λ-terms defined informally. Whereas the informal definition of the set of terms corresponds well with the formal definition, in the case of the set of λ-terms the situation is different. The symbol λ is frequently seen as an 'abstractor', but it is not all that clear if λ really has a general purpose capacity to abstract, or if operators in the underlying signature 'owns' their abstractions, i. e., is λ in an abstraction really to be indexed by the operator it abstracts. Note also that it is actually the operator that is abstracted to another operator, not the term that is abstracted to an operator. Traditional presentations of λ-calculus is not always specific on that point. Church [28] called "λ" an *improper symbol*, together with "(" and ")" also being improper symbols. The *proper symbols* are then those residing in the signature, or being symbols for variables. Church's *simple typing* has as a motivation the statement that *a complete incorporation of the calculus of λ-conversion into the theory of types is impossible if we require that λ and juxtaposition shall retain their respective meanings as an abstraction operator and as denoting the application of function to*

argument [28]. Therefore λ is not to be seen as an operator in any signature. Church's types ι and o, and their algebras, are not at all obvious, and it is indeed unclear in which signature they actually reside. Church introduces the type constructor so that $(\beta\alpha)$ is a type whenever β and α are types.

The informal definition of untyped λ-terms is as follows:

1. a variable is a λ-term;
2. if M is a λ-term, then λx.M, with x a variable, is a λ-term (abstraction); and
3. if M and N are λ-terms, then M N is a λ-term (application).

This is seen as an elegant definition of λ-term, even if it is unspecific, e. g., about possible underlying 'basic' terms formed out of an underlying signature. Variables are such basic terms, but it is not explicitly said that all underlying basic terms are λ-terms. If the underlying signature is $\Sigma = (S, \emptyset)$, then variables are the only basic terms. Usually basic terms are seen as λ-terms also in the case where $\Omega \neq \emptyset$. For example, for the basic term $x + 1$, $\lambda x.x + 1$ is the abstraction.

The situation regarding syntactically constructed natural numbers is interesting. The natural numbers can be introduced as λ-terms either as basic terms given the NAT signature, or, as Church did, fixing a function variable $f_{\alpha\alpha}$ essentially as a 'fixsymbol' for natural numbers, and then proceed to define

$$0_{\alpha'} = \lambda f_{\alpha\alpha}.\lambda x_{\alpha}.x_{\alpha},$$
$$1_{\alpha'} = \lambda f_{\alpha\alpha}.\lambda x_{\alpha}.(f_{\alpha\alpha}x_{\alpha}),$$
$$2_{\alpha'} = \lambda f_{\alpha\alpha}.\lambda x_{\alpha}.(f_{\alpha\alpha}(f_{\alpha\alpha}x_{\alpha})),$$

and so on.

Now, because a typed calculus relies on type terms we must separate the terms of the type level and the value level. These levels will therefore consist of two separate many-sorted signatures. To introduce primitive operations in our language – such natural numbers according to the NAT signature – we also need a signature for this language of "primitive sorts and operations".

In total, we will therefore have three signatures and, as a consequence, three notions of terms. As will become clear, these signatures are dependent on each other in a natural way. Specifically, the type level signature is necessary to express the value level signature and the primitive level signature is necessary to express both the type level and value level signatures.

Thus, the underlying basic signatures do not include function types and a categorical handling, e. g., of λ-term functors therefore requires an arrangement into levels of signatures. Consider two sorts $\mathtt{s}_1$ and $\mathtt{s}_2$ in some set of sorts S. The function sort involving $\mathtt{s}_1$ and $\mathtt{s}_2$ can be denoted $\mathtt{s}_1 \Rrightarrow \mathtt{s}_2$. Even if we want to view $\mathtt{s}_1 \Rrightarrow \mathtt{s}_2$ as a (constructed) sort, it is not part of S. The question is then how to expand the signature $\Sigma = (S, \Omega)$ to a signature $\Sigma' = (S', \Omega')$ so that $\mathtt{s}_1 \Rrightarrow \mathtt{s}_2 \in \mathtt{S}'$ whenever $\mathtt{s}_1, \mathtt{s}_2 \in \mathtt{S}$. Such an arrangement also enables us to keep λ-abstractions, as members of Ω', clearly apart from λ-terms, residing in the set of λ-terms as defined by the λ-term functor.

Our three-level arrangement of signatures then have the basic signature Σ on level one, while Σ' is on level three. On level two we have the (Σ-)*superseding type signature* as a one-sorted signature $\mathtt{S}_\Sigma = (\{\mathtt{type}\}, Q)$, where Q is a set of *type constructors* satisfying

1. $\mathtt{s}\colon \to \mathtt{type}$ is in Q for all $\mathtt{s} \in \mathtt{S}$
2. there is an operation $\Rrightarrow\colon \mathtt{type} \times \mathtt{type} \to \mathtt{type}$ in Q.

If Q does not contain any other type constructors, apart from those given by 1 and 2, we say that $\mathtt{S}_\Sigma$ is a (Σ-)*superseding simple type signature*.

For any Σ-superseding type signature $\mathtt{S}_\Sigma$ we obviously have the term monad, called the *type term monad*, $\mathsf{T}_{\mathtt{S}_\Sigma}$, over a given category whose objects represents the variables that on the superseding level intuitively correspond to the notion of type variables. Given this interpretation then $\mathsf{T}_{\mathtt{S}_\Sigma} X$, where X is the object representing type variables, contains all terms that we call *type terms*. We may write $\mathtt{s} \Rrightarrow \mathtt{t}$ for the type term $\Rrightarrow(\mathtt{s}, \mathtt{t})$, which we call an *arrow type term*.

The signature $\Sigma' = (S', \Omega')$ on level three then is based on $S' = \mathsf{T}_{\mathtt{S}_\Sigma}\emptyset$, i. e., the sorts on level three are those from level one

together with the constructed sorts, on level two appearing as terms (the type terms), added to those basic sorts coming from level one.

For the operators in Ω' it may seem natural to include all operators from Ω into Ω' so that $\Omega \subseteq \Omega'$, but it is not always desirable. If we consider the NAT signature on level one we obviously may have both $\mathtt{0}\colon \to \mathtt{nat}$ and $\mathtt{succ}\colon \mathtt{nat} \to \mathtt{nat}$ included in the operators for $\mathtt{NAT}'$. However, the unary operator succ, i. e., unary both on level one and level three, can alternatively be abstracted to become a constant (0-ary) operator $\lambda^{\mathtt{succ}}\colon \to (\mathtt{nat} \Rrightarrow \mathtt{nat})$ on level three. Clearly, the constant $\mathtt{0}\colon \to \mathtt{nat}$ converts to $\lambda^{\mathtt{0}}\colon \to \mathtt{nat}$, i. e., a constant on level remains as a constant on level three. Note also that nat on level one is not the same as nat on level three. If we need to be strict, we should use e. g., $\mathtt{nat}'$ for the corresponding sort on level three.

We usually want to have $S \subseteq S'$, i. e., an S is embedded into S', where nat maps to $\mathtt{nat}'$, but not necessarily $\Omega \subseteq \Omega'$. The situation involving the construction of λ-terms, where $\Omega \not\subseteq \Omega'$, is discussed more in detail in the subsection below.

Remark. For $\Sigma = (S, \Omega)$ on level one, $\mathtt{S}_\Sigma$ can obviously be extended with further operators beyond just $\Rrightarrow$. It may, for example, be of interest to include additional type constructors to, e. g., introduce products $*\colon \mathtt{type} \times \mathtt{type} \to \mathtt{type}$ or include type transformers such as $\mathtt{F}\colon \mathtt{type} \to \mathtt{type}$ intuitively corresponding to a functor. This will be applied in Section 5.2 below for description logic, were we use the powerset functor to represent relations.

Remark. Note, in the mathematical notation for signatures according to Bénabou [19], that the monoidal arrangement does not embrace any structure related to the Σ-superseding level. That is, in Bénabou [19] there is no clearly expressed anticipation of applications in type theory.

Remark. Whereas the algebras $\mathfrak{A}(\Sigma)$ of signatures Σ, involving assignments of sorts $\mathtt{s}$ to domains $\mathfrak{A}(\mathtt{s})$ of $\mathfrak{A}$, are standard according to universal algebra, the 'algebra' of the (Σ-)superseding type signature $\mathtt{S}_\Sigma$ is not immediate since the domain assigned to the sort type clearly cannot be just a set. There are several options for this, and these considerations may go beyond traditional uni-

versal algebra. We will here consider the issue of semantics only superficially and leave a deeper treatment for future publications.

Church's type constructor correspond to our $\Rrightarrow$, so that our $(\beta \Rrightarrow \alpha)$ is his $(\beta\alpha)$. An interpretation of Church's ι to be our `type` is clearly not controversial, but for the interpretation of o there are a number of alternative intuitions. Church says, o *is the type of propositions*, but at that point nothing is said about *proposition*. Indeed, Church states the following:

> We purposely refrain from making more definite the nature of the types o and ι, the formal theory admitting of a variety of interpretations in this regard. Of course the matter of interpretation is in any case irrelevant to the abstract construction of the theory, and indeed other and quite different interpretations are possible (formal consistency assumed).
>
> – Church [28]

We will conclude this section with a discussion on natural language expressions involving modifiers and quantifiers, and the possibility to understand and even encode these expressions more formally. Lotfi Zadeh frequently uses expressions such as "most Swedes are tall" and "there are more small balls than large balls in the box". The difficulty in handling these expressions in fuzzy logic is well-known, and unique solutions have not been presented in the literature. We believe that unique solutions in fact do not exist, and here we will provide a brief view about possibly encodings of such expression, or related subexpressions, in our three level exposition of signatures involving type constructors. Intuitively it seems as modifiers are closely related to type constructors, or at least that modifiers are operators on level three being specified using constructed types on level two. Quantifiers are more like abstractors of sentences, and it may be anticipated that the formalization of quantifiers in this sense is similar to the formalization of the way λ acts expressions.

Let us have a closer look at modifiers. When we say "most a are b" we usually see a is specified by types of non-Boolean character, whereas b is more seen as specified by a Boolean-like type. In this case "most" is about counting something we loosely

accept to be $\mathfrak{a}$'s that satisfy the condition that we semantically and individually understand to be $\mathfrak{b}$. This counting is more like a process or procedure, or almost like an algorithm producing a degree of confidence that indeed "most $\mathfrak{a}$ are $\mathfrak{b}$". It is, however, unclear whether or not "most" can be seen as a quantifier in some general sense. If we say "there are more $\mathfrak{a}$'s than $\mathfrak{b}$'s in $\mathfrak{c}$", or concretely, "there are more apples than pears in the fruit basket", this may be informative in the kitchen. Note also that "there are more apples than pears, but there are less Ingrid Marie apples than Anjou pears in the fruit basket" is maybe more informative for advanced cooking. An expert on fruit might also say "the Comice pear on the fruit plate looks more fresh than the Granny Smith apple". The *fruit plate* and *fruit basket* clearly shows we are dealing with sets and even hierarchies of sets, in a sense to be semantically defined by and within respective context. A basket can be seen as having more structure than a plate, and a plate is something more that just a set.

Before proceeding to make these examples more concrete, we introduce some basic prerequisites about type constructors and the superseding signatures. For any two unary type constructors

$$\phi, \psi \colon \mathtt{type} \to \mathtt{type},$$

we define the *composed type constructor*

$$\psi \circ \phi \colon \mathtt{type} \to \mathtt{type}$$

by

$$(\psi \circ \phi)\mathtt{s} = \psi(\phi \mathtt{s}).$$

For unary type constructors $\phi, \psi \colon \mathtt{type} \to \mathtt{type}$, a *type transformation* τ *from* ϕ *to* ψ, denoted $\tau \colon \phi \Rightarrow \psi$, if it exists, is assumed, for all $\mathtt{s} \in \mathsf{T}_{\mathtt{s}_\Sigma}\emptyset$, to be given by a unique (constant operator) $\tau_{\mathtt{s}} \colon \to (\phi \mathtt{s} \Rrightarrow \psi \mathtt{s})$.

It is clearly tempting to view the type constructor ϕ itself as a functor, but syntactically it obviously not a functor. Further, we may want to assume that any $\mathtt{f} \colon \to (\mathtt{s} \Rrightarrow \mathtt{t})$ gives rise to a unique $\phi\mathtt{f} \colon \to (\phi\mathtt{s} \Rrightarrow \phi\mathtt{t})$, and this intuitively opens up the question e. g., preservation of composition, as we syntactically

have not defined any composition of constants, otherwise than using the application operator on level three.

Example 5.1.1. Hierarchies of sets, or sets of sets, sets of sets of sets, and so on, can be modelled by the 'powerset' type constructor $\mathsf{P}\colon \mathtt{type} \to \mathtt{type}$ on level two, i. e., intuitively thinking that P indeed is a powerset functor, not necessarily the ordinary powerset functor, over `Set`. We have to pay attention to 'double powerset' type constructors, since in the case of composing powerset functor into a so called double powerset functor we have two choices, namely, composing the covariant powerset functor with itself or the contravariant powerset functor with itself. Note that composing contravariant functors produces a covariant functor. We may denote the 'contravariant powerset' type constructor by $\overline{\mathsf{P}}\colon \mathtt{type} \to \mathtt{type}$. For the composition of the contravariant powerset type constructor with itself we would expect $\mathfrak{A}(\overline{\mathsf{P}} \circ \overline{\mathsf{P}}) = \mathfrak{A}(\overline{\mathsf{P}}) \circ \mathfrak{A}(\overline{\mathsf{P}})$.

We are now able to show how encoding a fruit basket and its content needs to make clear distinctions about what are types on level two and which are the constants or operators in general on level three. It seems natural to say that *Ingrid Marie* and *Anjou* are types of apples and pears, respectively, so we should then have $\mathtt{IngridMarie}, \mathtt{Anjou}\colon \to \mathtt{type}$ on level two. This enables to have a specific apple apple_0 and a pear pear_0 as constants on level three if we include $\mathrm{apple}_0\colon \to \mathtt{IngridMarie}$ and $\mathrm{pear}_0\colon \to \mathtt{Anjou}$, or we may declare variables x and y according to $x :: \mathtt{IngridMarie}$ and $y :: \mathtt{Anjou}$. Then, both substitutions $x := \mathrm{apple}_0$ and $y := \mathrm{pear}_0$ make sense.

We should clearly include $\mathtt{Fruit}, \mathtt{Apple}, \mathtt{Pear}\colon \to \mathtt{type}$ on level two, and then we can include $\mathit{Apple}, \mathit{Pear}\colon \to \mathtt{Fruit}$ on level three, but Apple and Pear as (constant) terms on level three must not be identified or confused with `Apple` and `Pear` as terms on level two. Similarly, we can include $\mathrm{IngridMarie}\colon \to \mathtt{Apple}$ and $\mathrm{Anjou}\colon \to \mathtt{Pear}$ on level three, where again IngridMarie on level three and `IngridMarie` on level two, and Anjou on level three and `Anjou` on level two, must not be identified. On level three we are then allowed to included also that Apple and Pear are of type `Fruit`.

For the typing of the fruit basket we have a number of options. We may see fruit baskets as always allowing any fruits, or we may want to have different baskets for fruits in general, but also baskets specifically for apples, pears, and so on. For the latter, we would first specify $\texttt{FruitBasket}\colon \texttt{type} \to \texttt{type}$ on level two as a type intuitively being a powerset constructor, so that

$$\texttt{FruitBasket}(\texttt{Fruit}) :: \texttt{type},$$
$$\texttt{FruitBasket}(\texttt{Apple}) :: \texttt{type}, \text{ and}$$
$$\texttt{FruitBasket}(\texttt{Pear}) :: \texttt{type}$$

appear as terms on level two, and then include

$$\texttt{AllFruitsBasket}\colon \to \mathsf{P}(\texttt{Fruit}),$$
$$\texttt{AppleFruitBasket}\colon \to \mathsf{P}(\texttt{Apple}), \text{ and}$$
$$\texttt{PearFruitBasket}\colon \to \mathsf{P}(\texttt{Pear})$$

as a constant on level three. This captures the idea that a fruit basket is a set of fruits, i. e., $\text{FruitBasket} :: \mathsf{P}(\texttt{Fruit})$.

When we start to use the basic constants and terms in connection with other operators providing counting we must decide to have one or the other, and be very careful about not moving freely between respective definitions. In natural language, this in fact happens frequently, and one person communicating with another is not always aware of which typing is adopted. Even worse, if nobody is aware of distinct typing, each and everyone may shift to adopt definitions from one typing to another. This is the main problem of natural language.

Counting fruits, apples or pears in the fruit basket is now something close to establishing cardinality of subsets of apples and pears in a set of fruits, i. e., we might have operators

$$\text{card}^{\texttt{Apple}}\colon \texttt{FruitBasket}(\texttt{Fruit}) \to \texttt{nat} \text{ and}$$
$$\text{card}^{\texttt{Pear}}\colon \texttt{FruitBasket}(\texttt{Fruit}) \to \texttt{nat}$$

on level three. Here we must assume to have nat on level one so that $\texttt{nat}\colon \to \texttt{type}$ is on level two. Similarly we may have

$$\text{card}^{\texttt{IngridMarie}}\colon \texttt{FruitBasket}(\texttt{Apple}) \to \texttt{nat} \text{ and}$$
$$\text{card}^{\texttt{Anjou}}\colon \texttt{FruitBasket}(\texttt{Pear}) \to \texttt{nat}$$

for the cardinality of the subset of Ingrid Marie apples in a set of apples, and the cardinality of Anjou pears in a set of pears.

Finally we need to include conversion operators so that an Ingrid Marie really is an apple, an Anjou really a pear, and that apples and pears are fruits.

In summary, the typing on level two is

$$\begin{aligned} \mathsf{P}, \mathtt{FruitBasket}, \mathtt{FruitPlate} &: \mathtt{type} \to \mathtt{type} \\ \mathtt{Fruit}, \mathtt{Apple}, \mathtt{Pear} &: \to \mathtt{type} \\ \mathtt{IngridMarie}, \mathtt{Anjou} &: \to \mathtt{type} \\ \mathtt{nat} &: \to \mathtt{type} \end{aligned}$$

and on level three we have

$$\begin{aligned} \mathtt{AllFruitsBasket} &: \to \mathtt{FruitBasket}(\mathtt{Fruit}) \\ \mathtt{AppleFruitBasket} &: \to \mathtt{FruitBasket}(\mathtt{Apple}) \\ \mathtt{PearFruitBasket} &: \to \mathtt{FruitBasket}(\mathtt{Pear}) \\ \mathrm{Apple}, \mathrm{Pear} &: \to \mathtt{Fruit} \\ \mathrm{IngridMarie} &: \to \mathtt{Apple} \\ \mathrm{Anjou} &: \to \mathtt{Pear} \\ \mathrm{apple}_0, \mathrm{apple}_1, \ldots &: \to \mathtt{IngridMarie} \\ \mathrm{pear}_0, \mathrm{pear}_1, \ldots &: \to \mathtt{Anjou} \end{aligned}$$

as constants,

$$\begin{aligned} \mathrm{card}^{\mathtt{Apple}} &: \mathtt{FruitBasket}(\mathtt{Fruit}) \to \mathtt{nat} \\ \mathrm{card}^{\mathtt{Pear}} &: \mathtt{FruitBasket}(\mathtt{Fruit}) \to \mathtt{nat} \\ \mathrm{card}^{\mathtt{IngridMarie}} &: \mathsf{P}(\mathtt{Apple}) \to \mathtt{nat} \\ \mathrm{card}^{\mathtt{Anjou}} &: \mathsf{P}(\mathtt{Pear}) \to \mathtt{nat} \end{aligned}$$

as the previously described cardinality operators, and

$$\begin{aligned} \phi^{\mathtt{Apple} \to \mathtt{Fruit}} &: \mathtt{Apple} \to \mathtt{Fruit} \\ \phi^{\mathtt{Pear} \to \mathtt{Fruit}} &: \mathtt{Pear} \to \mathtt{Fruit} \\ \phi^{\mathtt{IngridMarie} \to \mathtt{Apple}} &: \mathtt{IngridMarie} \to \mathtt{Apple} \\ \phi^{\mathtt{Anjou} \to \mathtt{Pear}} &: \mathtt{Anjou} \to \mathtt{Pear} \end{aligned}$$

as the obvious injection operators.

We now leave it to reader to extend the example with various definitions for "there are more apples than pears, but there are less Ingrid Marie apples than Anjou pears in the fruit basket", where at least types and operators from `NATORD` need to be included at level one. This may expectedly be done in a number of ways.

An additional subtlety arises from the intuitively different semantics of 'set', 'plate', and 'basket' that then also calls for using type transformations between `FruitBasket` and P, and between `FruitPlate` and P, semantically acting as a forgetful or flattening transformations

$$\tau^{\text{BasketToSet}}_{\texttt{Fruit}} \colon \ \to (\texttt{FruitBasket}(\texttt{Fruit}) \Rrightarrow \mathsf{P}(\texttt{Fruit}))$$
$$\tau^{\text{PlateToSet}}_{\texttt{Fruit}} \colon \ \to (\texttt{FruitPlate}(\texttt{Fruit}) \Rrightarrow \mathsf{P}(\texttt{Fruit})).$$

In this context, note how a corresponding type transformation from `FruitBasket` to `FruitPlate`, or, similarly, from `FruitPlate` to `FruitBasket`, is much less obvious, as the former may be represented simply by a action pouring fruit from a plate into a basket, and the latter allow fruit from a basket to fall randomly out onto a plate.

This example can then be extended further to work over selections of underlying categories with $\texttt{Set}(\mathfrak{L})$ as the prime example for such an underlying category.

Let us now consider λ-terms and the possibility of fuzzy λ-calculus. Note, we will here consider simply typed lambda calculus using the combinatory approach introduced by Schönfinkel [124] and Curry [32]. The approach of using combinators will prove useful when going from a crisp notion of computation to a fuzzy one. Note, this combinatory method differs somewhat from that of Manzonetto and Salibra [103] who develop an algebraic treatment of untyped lambda calculus while retaining the notion of lambda abstractions that incorporate bound variables. To do so, they construct λ-terms from a custom notion of signature taking into account bound and free variables. As contrast, the combinatory approach is advantageous in that we may completely rely on the previously defined term monads that are in-

herently ignorant as to the significant difference between bound and free variables.

Thus, the three signature levels underlying our production of λ-terms are the following.

1. The level of primitive underlying operations, with a usual many-sorted signature $\Sigma = (S, \Omega)$

2. The level of type constructors, with a single-sorted signature

$$\begin{aligned} \mathsf{S}_\Sigma = \{&\{\mathtt{type}\},\\ &\{\mathsf{s}\colon \to \mathtt{type} \mid \mathsf{s} \in \mathsf{S}\} \cup\\ &\{\Rrightarrow\colon \mathtt{type} \times \mathtt{type} \to \mathtt{type}\}\\ \}& \end{aligned}$$

   As previously mentioned the $\Rrightarrow$ operation is typically written in an infix position, i. e., we write $\mathsf{s} \Rrightarrow \mathsf{t}$ instead of $\Rrightarrow(\mathsf{s}, \mathsf{t})$. Further, we let $\Rrightarrow$ be right associative, i. e., a type term $\mathsf{s} \Rrightarrow \mathsf{t} \Rrightarrow \mathsf{u}$ is interpreted as $\mathsf{s} \Rrightarrow (\mathsf{t} \Rrightarrow \mathsf{u})$.

3. The level including our combinators and λ-abstracted operations based on the signature $\Sigma' = (S', \Omega')$ where $S' = \mathsf{T}_{\mathsf{S}_\Sigma}\emptyset$ and

$$\begin{aligned} \Omega' = \{&\lambda^{\omega}\colon \to (\mathsf{s}_1 \Rrightarrow \cdots \Rrightarrow \mathsf{s}_n \Rrightarrow \mathsf{s})\\ &\qquad \mid \omega\colon \mathsf{s}_1 \times \cdots \times \mathsf{s}_n \to \mathsf{s} \in \Omega\} \cup\\ &\{\mathsf{I}_\mathsf{s}\colon \to (\mathsf{s} \Rrightarrow \mathsf{s}) \mid \mathsf{s} \in \mathsf{S}'\} \cup\\ &\{\mathsf{K}_{\mathsf{s},\mathsf{t}}\colon \to (\mathsf{s} \Rrightarrow \mathsf{t} \Rrightarrow \mathsf{s}) \mid \mathsf{s}, \mathsf{t} \in \mathsf{S}'\} \cup\\ &\{\mathsf{S}_{\mathsf{s},\mathsf{t},\mathsf{u}}\colon \to ((\mathsf{s} \Rrightarrow \mathsf{t} \Rrightarrow \mathsf{u}) \Rrightarrow (\mathsf{s} \Rrightarrow \mathsf{t}) \Rrightarrow \mathsf{s} \Rrightarrow \mathsf{u})\\ &\qquad \mid \mathsf{s}, \mathsf{t}, \mathsf{u} \in \mathsf{S}'\} \cup\\ &\{\mathtt{app}_{\mathsf{s},\mathsf{t}}\colon (\mathsf{s} \Rrightarrow \mathsf{t}) \times \mathsf{s} \to \mathsf{t} \mid \mathsf{s}, \mathsf{t} \in \mathsf{S}'\} \end{aligned}$$

   Note, the type subscripts of are often omitted and the $\mathtt{app}$ operator is often written using plain juxtaposition, i. e., instead of $\mathtt{app}(\lambda^{\mathtt{succ}}, x)$ we write $\lambda^{\mathtt{succ}}\, x$. Further, such juxtaposition is taken to be left associative, for example, the term $\mathsf{S}\,\mathsf{K}\,\lambda^{\mathtt{succ}}\,\lambda^{\mathtt{zero}}$ is interpreted as $((\mathsf{S}\,\mathsf{K})\,\lambda^{\mathtt{succ}})\,\lambda^{\mathtt{zero}}$.

Note what abstraction really and precisely is doing. Abstraction takes an operator in Ω (in Σ), 'waits' for the related sorts to 'move over' to constants in the superseding level signature, and for these constants as terms (on the superseding level) to 'move over' to appear as sorts in S' (in Σ'). A notable observation is then that we do not have anything like an 'abstraction of an abstraction'. Note also that we are indeed including $\lambda^{\mathtt{succ}}\colon \rightarrow (\mathtt{nat} \Rrightarrow \mathtt{nat})$ into Ω', but not something like $\kappa^{\mathtt{succ}}\colon \mathtt{nat} \rightarrow \mathtt{nat}$ into Ω'. It would be possible, but whereas $\lambda^{\mathtt{succ}}$ is an abstraction, $\kappa^{\mathtt{succ}}$ leaves succ basically as it appears in Ω.

Clearly, there are many possible semantic interpretations of types and operators in programming languages – for an overview see, e. g., Gunter [75] – but we may here consider a simple algebraic approach. This requires us to limit our language to primitive recursion, which will be done over natural numbers. For more elaborate semantics and unrestricted recursion it is necessary to establish the semantic aspects of the language over some richer structures as, e. g., introduced in domain theory [1, 126]. In our simplified view we first let $\mathfrak{A}_\Sigma$ and $\mathfrak{A}_{\Sigma'}$ be algebras for Σ and Σ', respectively. Assume that $\mathfrak{A}_\Sigma$ is such that, for example,

$$\mathfrak{A}_\Sigma(\mathtt{nat}) = \mathbb{N} \quad \text{and} \quad \mathfrak{A}_\Sigma(\mathtt{0}) = 0 \in \mathbb{N}.$$

Note how it would be a definition or choice to say that also

$$\mathfrak{A}_{\Sigma'}(\mathtt{nat}) = \mathfrak{A}_\Sigma(\mathtt{nat}) \quad \text{and} \quad \mathfrak{A}_{\Sigma'}(\lambda^{\mathtt{0}}) = \mathfrak{A}_\Sigma(\mathtt{0})$$

but clearly the algebras can be defined differently for the Σ and Σ' levels and their related terms. With the choice taken it seems natural now to consider

$$\mathfrak{A}_{S_\Sigma}(\mathtt{type}) = \mathrm{Ob}(\mathtt{Set})$$

and

$$\mathfrak{A}_{S_\Sigma}(\mathtt{s} \Rrightarrow \mathtt{t}) = \mathrm{Hom}(\mathfrak{A}_{S_\Sigma}(\mathtt{s}), \mathfrak{A}_{S_\Sigma}(\mathtt{t}))$$

giving

$$\mathfrak{A}_{\Sigma'}(\lambda^{\mathtt{succ}}) \in \mathrm{Hom}(\mathfrak{A}_{\Sigma'}(\mathtt{nat}), \mathfrak{A}_{\Sigma'}(\mathtt{nat})).$$

Concerning variable substitutions we now are lead to the observation mentioned earlier concerning free and bound variables. To start with, note that the renaming issue is really about being 'hygienic' – a word used, e. g., by Barendregt [13] – or careful when selecting free and bound variables so as to avoid variable overlap in the sense that a term with a free variable being included by substitution into another term leads to the free variable becoming bound. The definition of substitutions must then consider these distinctions, so that full 'hygiene' is achieved. For example, substituting x by $\mathsf{succ}\, y$ in $\lambda y.\mathsf{succ}\, x$ requires a renaming of the bound variable y, e. g., $\lambda z.\mathsf{succ}\,(\mathsf{succ}\, y)$. Such considerations, however, are not suitable for functorial treatments in which clear separation between free and bound variables are ineffectual.

In our combinatory approach, as mentioned, we by definition avoid the need for renaming and may use the conventional notion of variable substitution and may indeed use it equally both on level one and three. Note, to be clear, in the functorial approach substitutions are always and nothing but morphisms in the Kleisli category of the selected term monad. A substitution in a substitution logic over a signature Σ (regardless of where it appears), is thus a morphism $\sigma\colon X \longrightarrow \mathsf{T}_\Sigma Y$, where we may have $X = Y$, but must not have. This also means that

$$\mu_Y \circ \mathsf{T}_\Sigma \sigma\colon \mathsf{T}_\Sigma X \longrightarrow \mathsf{T}_\Sigma Y$$

is the morphism that transforms terms to terms, given the particular substitution σ. So for $t \in \mathsf{T}_\Sigma X$ we have $(\mu_Y \circ \mathsf{T}_\Sigma \sigma)(t)$ as the result of the overall substitution. Frequently, the literature provides notation for what happens when only one $x \in X$ is affected by a particular $\sigma\colon X \longrightarrow \mathsf{T}_\Sigma X$. In this case, the assumption is that $\sigma(y) = y$ whenever $y \neq x$. A typical notation for $(\mu_X \circ \mathsf{T}_\Sigma \sigma)(t)$ in this case is $t[x := \sigma(x)]$ or $t[\sigma(x)/x]$, frequently used in functional programming, or, $t\{x \longmapsto \sigma(x)\}$, more used in logic programming. The logic programming notation in a more general form, for any σ, is then $t\{x_1 \longmapsto \sigma(x_1), \ldots, x_n \longmapsto \sigma(x_n)\}$, meaning $\sigma(y) = y$, always when $y \neq x_i, i = 1, \ldots, n$, and in its most general form $t\{x \longmapsto \sigma(x) \mid x \in X\}$, which is the intuitive notation corresponding precisely to $(\mu_X \circ \mathsf{T}_\Sigma \sigma)(t)$.

For example, using $\Sigma = \mathtt{NAT}$, note that $\mathtt{succ}(x)$, $x \in X$, is a term on level one, i. e., a member of $\mathsf{T}_{\mathtt{NAT}}X$, but $\mathtt{succ}$ is not, so it does not make any sense to, e. g., write $\mathtt{succ}[\mathtt{0}/x]$, whereas $\mathtt{succ}(x)[\mathtt{0}/x] = \mathtt{succ}(\mathtt{0})$ and $\mathtt{succ}(y)[\mathtt{0}/x] = \mathtt{succ}(y)$, for $x, y \in X$. The situation on level three is equivalent. We have, for example, $\lambda^{\mathtt{succ}}\, x[\lambda^{\mathtt{0}}/x] = \lambda^{\mathtt{succ}}\, \lambda^{\mathtt{0}}$, and crucially we avoid renaming in $\lambda^{\mathtt{succ}}\, x[\lambda^{\mathtt{succ}}\, y/x] = \lambda^{\mathtt{succ}}\, \lambda^{\mathtt{succ}}\, y$. Indeed, we may make this more precise using a trivial proposition.

Proposition 5.1.1. *Let* t_1 *and* t_2 *be two terms in* $\mathsf{T}_{\Sigma'}X_{S'}$*, and let* $\sigma\colon X_{S'} \longrightarrow \mathsf{T}_{\Sigma'}X_{S'}$ *be a variable substitution. Then*

$$(\mu_{X_{S'}} \circ \mathsf{T}_{\Sigma'}\sigma)(\mathtt{app}_{s,t}(t_1, t_2)) = \mathtt{app}_{s,t}((\mu_{X_{S'}} \circ \mathsf{T}_{\Sigma'}\sigma)(t_1), (\mu_{X_{S'}} \circ \mathsf{T}_{\Sigma'}\sigma)(t_2)).$$

Proof. Immediate from the term monad construction. □

At this point we have the crisp set of λ-terms, given the term functor $\mathsf{T}_{\Sigma'}\colon \mathtt{Set}_{S'} \longrightarrow \mathtt{Set}_{S'}$. The sets of λ-terms with respect to each end sort $s' \in S'$ are then represented by respective sets $\mathsf{T}_{\Sigma',s'}(X_s)_{s\in S'}$.

Fuzzy λ-terms can now be introduced either using monad compositions, or allowing $\mathsf{T}_{\Sigma'}$ to be a functor over the underlying category $\mathtt{Set}(\mathfrak{L})$ as previously described.

In Novák [112], fuzzy λ-calculus is treated differently as λ-terms remain the classical ones, and are informally defined. The underlying signatures are also crisp. Fuzzy aspects appears not until λ-calculus is used as a basis for logic. This is presented similar to the approach in Church [28], i. e., levels of signatures are not considered. The underlying basic terms are crisp, i. e., arise from the many-sorted term functor over $\mathtt{Set}$, even if the functorial construction of terms is not explicitly mentioned. Fuzziness in Novák [112] is therefore an ad hoc construction applied on the crisp and traditional view of λ-terms.

As a concluding remark, it is easily noted in the above constructions that the λ symbol could be avoided completely in, e. g., $\lambda^{\mathtt{succ}}$. We could simply write $\mathtt{succ}$ for the term (and constant operator) within the Σ' signature, and at the same time always pay attention to make the necessary distinction from its origin $\mathtt{succ}$

being the operator within the Σ signature. However, we prefer to keep the letter λ as a guide to form intuition for those readers already aware of the power of λ-calculus and λ-abstractions.

5.2 DESCRIPTION LOGIC

Widening the scope of relational structures begins with the observation that the category $\mathtt{Rel}$ of sets and relations is isomorphic with the Kleisli category $\mathtt{Set}_{\mathbf{P}}$ of the powerset monad $\mathbf{P}$ over the category $\mathtt{Set}$ of sets and functions. As already mentioned, every relation $R \subseteq X \times X$ is representable by a mapping $\sigma_R \colon X \longrightarrow PX$, and every mapping $\sigma \colon X \longrightarrow PX$ is representable by a relation $R_\sigma \subseteq X \times X$. It is easily seen that $\sigma_{R_\sigma} = \sigma$ and $R_{\sigma_R} = R$.

Considering other monads $\mathbf{F}$ over $\mathtt{Set}$, and over other categories $\mathtt{C}$, for that matter, widens the concept of 'relation' significantly, viewing it more like a 'substitution', and moreover, going beyond just $\mathtt{Set}$ enables to analyze various attributions, like uncertainty and stochasticity, to be canonically integrated into terms and sentences under considerations in chosen syntax for a logic.

The history of modal logic syntax goes back to Clarence Lewis and his work [97] dating back to the time after publication of *Principia Mathematica* and at the footstep of *Hilbert's Lectures*. The history of modal logic semantics started with work by Rudolph Carnap in the 1940's and culminates with Saul Kripke's semantical analysis of modal logic [93]. Thereafter we witness a wide range of extensions to modal logic, e. g., in dynamic logic that was initiated by Pratt [117] based on his lecture notes at MIT in 1974, again inspired by Floyd–Hoare's logic for program description, also adopted by Dijkstra in his predicate transformer semantics. Dynamic logic also mirrors to concepts within languages in the sense of "automata and languages", related, e. g., to Chomsky's hierarchies, and also spilling over to Kleene algebras and analysis of program construction in that fashion.

Another historical branch leading to description logic starts with Ross Quillian's developments of his word concepts [119] in the realm of memory models, and based on his PhD thesis in 1966 [118]. This restricts it to "hold only denotative, factual information", and "not a person's plans for doing things", in the latter

exclusion referring to Jean Piaget's "schemata" [116]. Quillian is not explicitly saying whether "denotative" is syntax or semantics, but when taking also Quillian work as a historical background to description logic, "denotative" may be seen as embracing both "syntactic (de)notation", i. e., signature, as well as semantic denotation, either as such without syntax, or as a semantic assignment for syntactic elements. Piaget is weaker on the distinction between syntax and semantics. In his *The Psychology of Intelligence* [116], Piaget writes about "thought psychology" and the psychological nature of logical operations, saying that *How far a psychological explanation of intelligence is possible depends on the way in which logical operations are interpreted: are they the reflection of an already formed reality or the expression of a genuine activity?* It may seem that Piaget on "already formed reality" explicitly refers to logical semantics, and by "expression of a genuine activity" to syntax, but this is not so. In fact, Piaget continues *the logician then proceeds as does the geometer with the space that he constructs deductively, while the psychologist can be likened to the physicist, who measures space in the real world.* Piaget almost wants to exclude logic from psychology, not include it.

Marvin Minsky introduced a knowledge representation schema with rule-based and logic-based formalisms [107]. In order to represent these concepts he introduced "frames", including descriptive information about how to understand and use frames. Collections of such frames are organized in systems in which the frames are interconnected.

Developments then pass through description languages, and eventually "language" is changed to "logic". *Description logic* (DL) is not a specific logic but rather seen as a family of logics being variants of the *attributive concept description language* ($\mathcal{ALC}$) [123], in turn based on the underlying *Formal Frame Description Language* ($\mathcal{FL}$) [23] for KL-ONE [24]. In Brachman and Levesque [23], *structured types* are mentioned in connection with frames [107], but typing here is not explained in more formal type theoretic fashion. Types become connected with concepts, and are treated similarly and intuitively as Quillian's (*nodes that can be reached by an exhaustive tracing process, originating at its initial*) *patriarchical type nodes* (*together with the total sum of relation-*

ships among these nodes specified by within-plane, token-to-token links) that are also interesting from typing point of view, even if Quillian's work does not touch upon typing or type constructors more formally, so there are no historical remarks on distinctions to be found for concepts and concept types. Roles and relations were at that time intuitively seen as relations between concepts, and not relations between their respective types.

We can now show how to connect modal operators, having relations as their semantics, to type constructors having powerset functors as their semantics. We will make these situations explicit in particular for description logics and its "existential roles" as modal operators. Typing the operators and quantifiers in this way also emphasizes the propositional nature of description logic.

For the purpose of this section we will refer to notations in Schmidt-Schauß and Smolka [123], and transform that machinery into our categorical framework for superseding signatures and related typing, eventually arriving at our enriched notions and properties for DL.

We will start by providing some required formalism to $\mathcal{ALC}$, which exists implicitly, and is mostly seen as folklore for the DL community. Nevertheless, it needs to be stated formally, before we can proceed with our categorical and term monadic formalism.

Firstly, we will examine the *interpretation* $\mathcal{I} = (\mathsf{D}^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\cdot^{\mathcal{I}}$ *maps every concept description* to a subset of $\mathsf{D}^{\mathcal{I}}$. The use of the letter D for that universe is a bit unfortunate, since D is also used for concept descriptions, e. g., in expressions like $\mathsf{C} \bigsqcup \mathsf{D}$, where D is not to be understood as the "D in $\mathsf{D}^{\mathcal{I}}$".

If C is a "concept", the interpretation $\mathsf{C}^{\mathcal{I}}$ of that concept is a subset of $\mathsf{D}^{\mathcal{I}}$, i. e., an element of the powerset $\mathsf{PD}^{\mathcal{I}}$. This means that $\mathsf{PD}^{\mathcal{I}}$ is the actual domain in the classical sense of domains as part of universal algebras. "Roles" R as relations $\mathsf{R}^{\mathcal{I}}$ on the semantic side are then seen as subsets of $\mathsf{D}^{\mathcal{I}} \times \mathsf{D}^{\mathcal{I}}$. This is equivalent to saying that $\mathsf{R}^{\mathcal{I}}$ is a mapping from $\mathsf{D}^{\mathcal{I}}$ to $\mathsf{PD}^{\mathcal{I}}$, which categorically is a morphism in the corresponding Kleisli category of the powerset monad $\mathbf{P} = (\mathsf{P}, \eta, \mu)$. The morphism $\mathsf{R}^{\mathcal{I}} \colon \mathsf{D}^{\mathcal{I}} \longrightarrow \mathsf{PD}^{\mathcal{I}}$ can be lifted to the morphism $\mathsf{PR}^{\mathcal{I}} \colon \mathsf{PD}^{\mathcal{I}} \longrightarrow \mathsf{PPD}^{\mathcal{I}}$. The inverse relation

R^{-1} is important in these respects as it is the actual relation used on the semantic side. It is easy to see that in fact

$$\begin{aligned}(\exists R : C)^{\mathcal{J}} &= \{a \in D^{\mathcal{J}} \mid \exists (a, b) \in R^{\mathcal{J}} : b \in C^{\mathcal{J}}\} \\ &= \mu_{D^{\mathcal{J}}}(PR^{-1}C)\end{aligned}$$

Now we should be aware of related typing and variables selected as being of expected types. In Schmidt-Schauß and Smolka [123], this situation is unclear, as they *assume the existence of two further disjoint alphabets of symbols* that are called *individual* and *concept variables*. In signatures and terms over signatures, variables are not part of the alphabet, in the sense of "alphabet" being the signature of sorts and operators. As before, if $\Sigma = (S, \Omega)$ is a (many-sorted) signature in the computer science notation, then variables are specific terms and are always typed. Categorically, we have the category $\mathtt{Set}_S$, and a set of variables is an object $(X_s)_{s \in S}$ in that category, and $T_\Sigma \colon \mathtt{Set}_S \longrightarrow \mathtt{Set}_S$ is the many-sorted term monad as described in Section 2.7.

If we speak of "individual concept" rather than "individual variable" we may use x, y, z as variables for individual concepts, and X, Y, Z as variables for concepts. It then remains to decide how to type "concept" and "individual concept", and here we will need type constructors. We will again change the language adopted in Schmidt-Schauß and Smolka [123] by saying "concept" instead of "individual concept", and "powerconcept" instead of "concept".

The key solution in this typing is that `concept` is a sort in the underlying given signature (on level one), and then we need to construct a *superseding* signature, so that `concept` becomes a constant operator in that signature. A type constructor P is then used to produce a new type `Pconcept`, which in its algebra will be assigned according to $\mathfrak{A}(\mathtt{concept}) = D^{\mathcal{J}}$ and $\mathfrak{A}(P(\mathtt{concept})) = PD^{\mathcal{J}}$.

We are now in position to describe the *Simply Typed Description Logic*. Note that the "existential quantifier" in the syntactic expression $\exists R : C$ is actually more like an "R-modality" applied to the powerconcept C, and that the semantic expression $(\exists R : C)^{\mathcal{J}}$ invokes the existential quantifier residing in the first-order logic for ZFC. In the following we will show this more explicitly, i. e.,

that this "existential quantifier" is more of a modal operator, and that such modal operators need not necessarily be equipped with semantics consisting only of relations à la Tarski.

Now let $\Sigma = (S, \Omega)$ be a signature, where $S = \{\mathtt{concept}\}$. We may add constants like $\mathtt{c}_1, \ldots, \mathtt{c}_n \colon \to \mathtt{concept}$ already at this level, and it would be tempting to call these "syntactic concepts" with $\mathfrak{A}_\Sigma(\mathtt{c}_i) \in \mathfrak{A}_\Sigma(\mathtt{concept})$, where $\mathfrak{A}_\Sigma(\mathtt{concept})$ could be equal to $D^\mathcal{J}$. However, we do not want to do that, as concepts and powerconcepts must reside in the same signature. We therefore go the superseding signature S_Σ, so that $\mathtt{concept} \colon \to \mathtt{type}$ becomes a constant in S_Σ. We now propose to include into S_Σ a type constructor $\mathsf{P} \colon \mathtt{type} \to \mathtt{type}$, with an intuitive semantics of being the powerset functor. Clearly, $\mathsf{P}(\mathtt{concept})$ is now the constructed type for "powerconcept". Note that $\mathsf{P}(\mathtt{concept})$ is a term on the Σ-superseding level, and a sort on S', so as a candidate for its algebra we could consider

$$\begin{aligned}\mathfrak{A}_{\Sigma'}(\mathsf{P}(\mathtt{concept})) &= \mathsf{P}\mathfrak{A}_{\Sigma'}(\mathtt{concept}) \\ &= \mathsf{P}D^\mathcal{J}.\end{aligned}$$

Note how we refrain from making the distinction between P as a syntactic and semantic object, and indeed we see this as a fundamental weakness of description logic, namely this intertwining of syntax and semantics has not been properly resolved within the description logic community. A variable $x \in X_{\mathsf{P}(\mathtt{concept})}$ is then a "concept variable" in the sense of Schmidt-Schauß and Smolka [123], and is also a 'term' in the sense of being an element of $\mathsf{T}_{\Sigma',\mathsf{P}(\mathtt{concept})}(X_s)_{s\in S'}$.

Now it is natural to include "constant concepts"

$$\mathtt{c}_1, \ldots, \mathtt{c}_n \colon \to \mathtt{concept},$$

with $\mathtt{concept}$ having been shifted, from being a term on the Σ-superseding level, over to becoming a sort in S'. Thus $\mathtt{c}_1, \ldots, \mathtt{c}_n$ are (constant) terms as elements of

$$\mathsf{T}_{\Sigma',\mathtt{concept}}(X_s)_{s\in S'},$$

i. e., $\mathtt{c}_1, \ldots, \mathtt{c}_n \in \mathsf{T}_{\Sigma',\mathtt{concept}}(X_s)_{s\in S'}$, and

$$\mathfrak{A}_{\Sigma'}(\mathtt{c}_i) \in \mathfrak{A}_{\Sigma'}(\mathtt{concept}) = D^\mathcal{J}.$$

Now "roles" are of the form

$$\mathtt{r}\colon \to (\mathsf{P}(\mathtt{concept}) \Rrightarrow \mathsf{P}(\mathsf{P}(\mathtt{concept}))),$$

and we need further to include operators

$$\eta\colon \to (\mathtt{concept} \Rrightarrow \mathsf{P}(\mathtt{concept})) \text{ and}$$
$$\mu\colon \to (\mathsf{P}(\mathsf{P}(\mathtt{concept})) \Rrightarrow \mathsf{P}(\mathtt{concept}))$$

into Ω'.

Note how a concept $\mathtt{c}\colon \to \mathtt{concept}$ is shifted to becoming a "one-point powerconcept" as represented by

$$\mathtt{app}_{\mathtt{concept},\mathsf{P}(\mathtt{concept})}(\eta, \mathtt{c}).$$

Now the syntactic expression like "$\exists r.x$", i. e., a term of type $\mathsf{P}(\mathtt{concept})$ can be defined according to

$$\exists r.x = \mathtt{app}_{\mathsf{P}(\mathsf{P}(\mathtt{concept})),\mathsf{P}(\mathtt{concept})}(\mu, \mathtt{app}_{\mathsf{P}(\mathtt{concept}),\mathsf{P}(\mathsf{P}(\mathtt{concept}))}(r, x))$$

and with the obvious algebras for η and μ, $\mathfrak{A}_{\Sigma'}(\exists r.x)$ will then be the expected one.

"Disjunction" and "negation" are then added as expected, i. e.,

$$\sqcup\colon \mathsf{P}(\mathtt{concept}) \times \mathsf{P}(\mathtt{concept}) \to \mathsf{P}(\mathtt{concept})$$

and

$$\neg\colon \mathsf{P}(\mathtt{concept}) \to \mathsf{P}(\mathtt{concept}),$$

with the obvious algebras, and "universal quantification" can be added accordingly.

It is now explicit how P as a type constructor at the Σ-superceding level, becomes the main building block for the "monadic modality operator" $\exists r$. With the semantics of P intuitively being the powerset functor (on the Σ-superceding level), we are more traditional, but nothing stops us from considering other set functors, or other endofunctors over more elaborate categories than just $\mathtt{Set}$.

Modeling uncertainty in this context provides an excellent example. The simply typed framework opens up many possibilities to define *fuzzy description logic*. In Yen [137] and Straccia [132] fuzzy DL is basically simply typed DL with the semantics of P intuitively being extended to the the many-valued powerset monad. It is really not "fuzzy logic for description" but rather something like "fuzzy description logic". A more canonic way to invoke uncertainty modeling is either to compose the many-valued powerset monad with the term monad $\mathbf{T}_{\Sigma'}$, or to allow the term functor $\mathsf{T}_{\Sigma'}$ to, e. g., go over $\mathtt{Set}(\mathfrak{L})$, where $\mathfrak{L}$ is a lattice.

We have shown how Simply Typed Description Logic, on the one hand, reveals the modal nature of DL more clearly, and, on the other hand, shows that the justification of speaking about the "existential quantifier" in DL is somewhat doubtful. A more precise meaning of DL being a "sublanguage of first-order logic" and expressions being "variable-free" is now also given in this typing framework. Signatures are very explicit and therefore syntax and semantics are kept apart more rigorously.

In these notations it is, given Example 5.1.1, also obvious how syntactic aspects of description logic can be extended to involve more elaborate relations than just the one represented by powerset functors. Indeed, double powerset functors, both covariant and contravariant can be used, as well as set functors for filters and ideals. Many other functors, e. g., representing relations in a generalized and broad sense can be used in this context, and their extension over various underlying categories can be investigated.

5.3 SOCIAL CHOICE

The discipline of social choice originates from objective probability used in justice and as pioneered by French mathematicians de Borda [33] and Condorcet [31]. The balance between individual liberty and societal authority was related to the risk of innocent citizens being wrongly convicted and punished for crime. Social justice as well as social order required that particular risk to be minimized. Condorcet argued that judicial tribunals could manage probabilities and errors, taking in to account also some minimum required plurality to guarantee the probability. Uncer-

tainty based voting schemas then are just behind the corner, and is the historical prerequisite also for choice theory.

Objective probability eventually turns subjective, and probabilists believe they have keys to inference mechanisms as well. Some modern time improvements can be seen in these directions, but generally speaking, probability is not logical.

The subject of social choice was revived in the 20th century by Arrow [6] who, facing the inconsistencies of group decisions, put the discipline of social choice in a structured axiomatic framework leading to the birth of social choice theory in its modern form. As Sen [128] pointed out "Arrow's impossibility theorem is a result of breathtaking elegance and power, which showed that even some very mild conditions of reasonableness could not be simultaneously satisfied by any social choice procedure, within a very wide family". Accordingly, impossibility results in social choice theory have been seldom considered as being destructive of the possibility of social choice and welfare economics. Sen [128] argued against that view, claiming that formal reasoning about postulated axioms, as well as informal understanding of values and norms, both point in the productive direction of overcoming social choice pessimism and of avoiding impossibilities.

Arrow's focused on individual values and ranking, together with impossibility theorems, and dealt with individual preferences and choice processes. Probabilities are not in the ingredients, but rather operators and functions, and properties about them. There are no counterparts in probability theory for these concepts. From a logical point of view, Arrow uses implicitly underlying signatures, even if they are never formalized, and since they aren't formalized, it is never seen that that these choice functions indeed could have been integrated into a logical framework. Arrow follows von Neumann and Morgenstern's "mathematical tradition" [111] in his success stories of economic and social sciences, but also without ending up in any logical framework.

In Eklund et al. [54] we assumed that making a distinction between choice and mechanism for choice could advantageously enrich the theoretical framework of social choice theory opening the way to categorical approaches. The idea of generalizing the Arrow's paradigm through a new architecture of social choice

procedure was introduced, e. g., by Bandyopadhyay [11] and then extended in Bandyopadhyay [12] where a social choice procedure is proposed which depends both on the way a set of alternatives is broken up into the subsets and the sequence in which each of these subsets is taken up for consideration.

Our standpoint here is that social choice functions must identify the difference between "we choose" and "our choice", the former being the operation of choosing, the latter being the result of that operation. We view this from a signature point of view, i. e., using formalism involving signatures and their algebras. Here we will consider signatures in the classical sense, i. e., a signature $\Sigma = (S, \Omega)$ consists of sorts, or types, in a set S, and operators in a set Ω.

In this algebraic formalism, an n-ary operator $\omega \in \Omega$ corresponds to the operation of choosing, and $\omega(t_1, \ldots, t_n)$ is a result of choosing, i. e., a choice. Note that both the operator ω as well as the term $\omega(t_1, \ldots, t_n)$ are syntactic representations of mechanisms for choosing and choices. The semantics of ω is a mapping $A(\omega)\colon A(s_1) \times \cdots \times A(s_n) \longrightarrow A(s)$.

Social choice is basically seen as a mapping

$$f\colon X_1 \times \cdots \times X_n \longrightarrow X$$

where agents $i \in \{1, \ldots, n\}$ are choosing or arranging elements in sets X_i. The aggregated social choice related to $x_i \in X_i, i = 1, \ldots, n$ is then represented by $f(x_1, \ldots, x_n)$. In most cases $X_1 = \cdots = X_n = X$, and the social choice function is then

$$f\colon X \times \cdots \times X \longrightarrow X.$$

This can be seen either as a semantic representation which has an underlying choice operator in its signature, or it is syntactic and elements in X are basically constant operators, i. e., $X = \Omega_0$ in some operator domain.

In the view of 'choosing' we would replace X with the set of substitutions. More precisely, let C be the Kleisli category $\mathtt{Set}_{\mathsf{T}_\Omega}$, where T_Ω is the term monad over $\mathtt{Set}$. Elements σ in $\mathrm{Hom}_{\mathsf{C}}(X, X)$ are then substitutions $\sigma\colon X \longrightarrow \mathsf{T}_\Omega X$, and $\mathcal{X} = \mathrm{Hom}_{\mathsf{C}}(X, X)$ is the

corresponding set of substitutions capturing the notion of individual choice and choosing. The choice function

$$\varphi\colon \mathcal{X} \times \cdots \times \mathcal{X} \longrightarrow \mathcal{X}$$

therefore may consider and compute with not just the output, the choice, but also with all the operators, i. e., the whole mechanism of choosing, leading to that particular term.

We will expand these ideas to cover uncertainty modelling, and we will show how representation of uncertainty can be seen as related to an appropriate choice of an underlying category. Furthermore, we will see how all this can be embedded into a many-sorted framework.

In the literature there are previous categorical approaches and the objective of Keiding [84] is similar to ours, namely a unification of framework, and indeed unification of concept, results and theorem framework based on more or less formal methods. Keiding involves categories and Hom functors, but the categorical framework remains rather poor, as there is no use of operators. The set $\mathrm{Hom}(A, PX)$ indeed comes with no structure. It is simply a set of mappings. In the end, we will have a Hom_C functor, where C also can carry uncertainty once (many-sorted) term monads are constructed over Goguen's category $\mathtt{Set}(\mathfrak{L})$. It then integrates both operators and uncertainties, and even more so, operators working internally over uncertainties. Eliaz [58], making no reference to Keiding [84], does not add any new formalism or formal methodology.

Arrow [6] studied social welfare functions, the arguments of which are named components of social states. These functions map n-tuples of individual preferences (orderings [6]) into a collective preference:

$$f\colon (X^m)^n \longrightarrow X^m$$

Here the assumption is that X is an ordering $(X, \preceq)$ with suitable properties. The preference value in this case is an ordinal value and not a scale value. Clearly, choice functions can also involve scale values, so that

$$f\colon (\mathbb{R}^m)^n \longrightarrow \mathbb{R}^m,$$

i. e., using the real line, or some suitable closed interval within the real line, for the preference (scale) values. Note how the underlying signature handles this situation internally for $\mathcal{X} = \mathrm{Hom}_{\mathbf{C}}(X, X)$, where $\mathbf{C}$ is the Kleisli category $\mathtt{Set}(\mathfrak{L})_{\mathbf{T}_\Sigma}$.

Computing with preferences is less transparent with orderings built into the set X of alternatives [54]. Also in this case there is a corresponding underlying signature capturing this situation.

In the presentation above we still use only terms. Sentences, satisfaction $\models$ (based on the algebraic models of the signature), and entailment $\vdash$ are not yet included. Axioms of the logic and inference rules for entailment are then also missing, so we have no "logic of choice" at this point. Clearly, this remains as future work.

Going beyond the distinction between choosing and choice, and entering rationality of choice, Mill [106] said that behaviour is based on custom more than rationality. Custom is clearly based on particular algebras acting as models and used in $\models$, whereas rationality is based on representable sentences interrelated by $\vdash$. These aspects are investigated in future work.

In consensus reaching [49, 51] there is a dynamic situation of aggregated choice, where individual preferences change within a consensus reaching mechanism. This opens up interesting perspectives as consensus reaching in our substitution model for social choice now also reaches the level of 'choosing', i. e., consensus is reached either on 'choice' level including dynamics for the 'choosing' level, or can even be a stronger consensus on 'choosing' levels as well. Similar situations appear in negotiation.

6

INFORMATION ONTOLOGY AND PROCESS STRUCTURE IN SOCIAL AND HEALTH CARE

The content of this chapter is for the most part reproduced from a paper submitted to the BPM 2012 conference.

Information ontology is mostly relational so epistemological enhancements are required as soon as we want to work logically with ontology. Description logic is not sufficient for these purposes since as a logic it is too poor. In health and social care it has been acknowledged that *information and process* should go hand in hand, so that whenever information or knowledge management system modules are used, we should always be aware of the 'location' in the process where this information is consumed, e. g., for documentation purposes or knowledge is applied in decision-making.

Whereas information has been in focus even with less logical considerations, process structures have been very little subject to ontology and logical investigation. BPMN[1] [113, 114] has grown to become almost a de facto standard for process modelling, and its BPEL counterpart is also quite a useful support for software development of underlying computational processes.

The weakness of BPMN is clearly its lack of semantics. Data Objects and 'tokens' are typical where no semantics whatsoever is provided. For Data Objects, the application developer is invited and is free to provide specific semantics, and as Data Objects are part of the syntax, it is a feasible task to do that. However, for tokens the situation is less clear, and this makes it rather difficult to suggest the way in which way tokens carry data.

In this chapter we view BPMN logically, and even if we do not provide a full logical syntax for BPMN syntax, we do suggest that Data Objects syntactically are terms in a logic. This means we see terms as provided by an underlying signature, and the

1 We will here use BPMN version 1.2 since we have used Microsoft's Visio 2010 to encode the melanoma care process.

natural view of tokens is then to see them as embracing variable substitution. We define terms very strictly within category theory as term monads, so that variable substitutions are morphisms in the Kleisli category over that selected term monad. So we say that a token embraces a Kleisli morphism. We also use the concept of superceding signatures, by which type constructors play an important role. This also in some sense means we are contributing to a typing of BPMN.

A further purpose of this chapter is to provide steps towards improving the situation by describing an information representation formalism which enables management of uncertainty as well as information composition in cross-functional scenarios. This typically happens on token level, and require to include aspects of Kleene algebras as represented by Hom-sets (sets of tokens) given that Kleisli. This can be seen as a Kleene algebra of tokens.

We illuminate our information, processes and logic using the melanoma care process and its underlying guidelines.

6.1 ONTOLOGY IN HEALTH AND SOCIAL CARE

Ontology in the health and social care domain relate to standards like HL7 and SNOMED CT. HL7 is more UML related and oriented towards databases and patient records, whereas SNOMED is seen more as an ontology in the "web ontology" sense. Even more so, SNOMED has adopted description logic as possible logical framework for its ontology. This can be seen as rather doubtful, since it is far from clear that health ontology relies logically on the same foundations as web ontology. Web and health ontology have different objectives, and will for sure take different development paths in the future. However, this is the present situation concerning SNOMED and logic. This chapter can be seen as an indication about "what SNOMED is and what SNOMED isn't", or even about "what description logic is and what description logic isn't".

Web ontology is only partially logical in that even the underlying signature for encoding vocabularies is treated informally with concepts being more like atoms in the sense of constant (o-

ary) operators. It's even unclear if relations, like IS_A in SNOMED CT, e. g., in statements like

```
open fracture of foot IS_A fracture of foot
                      IS_A injury of foot
                      IS_A disorder of foot
```

are considered logically as terms or sentences. They are intuitively in health ontology viewed as statements, i. e., sentences, but a closer logical inspection in Eklund et al. [57] reveal them as terms.

It should also be noted that description logic is not a formal logic as described in Eklund et al. [57]. Description logic does not even have a formal involvement of signature as it uses *concept* as a primitive notion. Concepts are used as terms and sentences are relational only, which means description logics is usually seen as kind of an informal subset of first-order logic. In Eklund et al. [57] we show it is more like a propositional logic since the existential quantifier is superficial in the syntax. Description logic therefore has severe difficulties to include reasoning mechanisms, which means that this logic as a partial ontology remains on the logic levels including signatures, terms and sentences only, and even being rather informal about them.

6.2 MELANOMA

From information and process point of view, care related to a specific cancer type can roughly be seen as initiated by the reasons for the patient to seek help. This is typically a voluntary visit to a primary care center, or a care process initiated by a emergency situation. It may also be a care process initiated or being ongoing for other reasons, where additional reasons lead to a cancer investigation. The process associated with initial seeking of care is illustrated in Figure 4.

Cancer involves tumors, their removal, and pathological examination of tissue (histology) is more common than cell examination (cytology). Pathology report are well standardized with respect to content and data formats, but less so with respect to

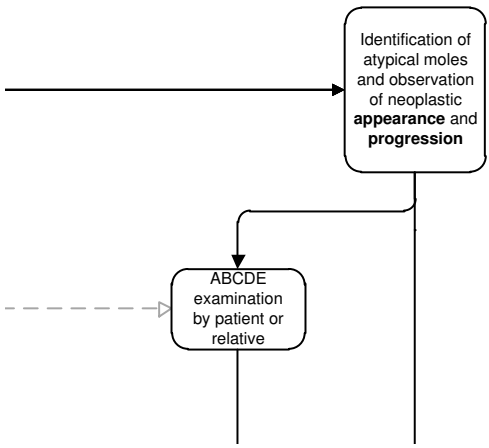


Figure 4: Initial seeking of care by patient.

document formats and structures. This means that electronic referrals and interoperability require proper encoding of these reports.

Common to all cancer types is also the staging and classifications such as provided, e. g., by the *TNM staging* from AJCC (Americal Joint Committee on Cancer). The TNM staging and prediction of survival rates from initial diagnosis is based on an enormous number of publications about cancer treatment in general including estimations of survival rates [29], and further as given for specific cancer types as given, e. g., in the Cochrane review on surgical excision margins for primary cutaneous melanoma [129].

Guidelines for cancer treatment are typically clinical guidelines made by and for clinicians, such as the SIGN (Scottish Intercollegiate Guidelines Network) guideline for cutaneous melanoma [127]. This guidelines is very detailed on all aspects of melanoma care focusing on intervention and treatment taking place in primary care and a secondary care dermatology, with treatment and follow-up focusing on cancer stages I, II and III. Pathology and surgery aspects are not core to this guideline, and aspects involving other stakeholders in health care and health information

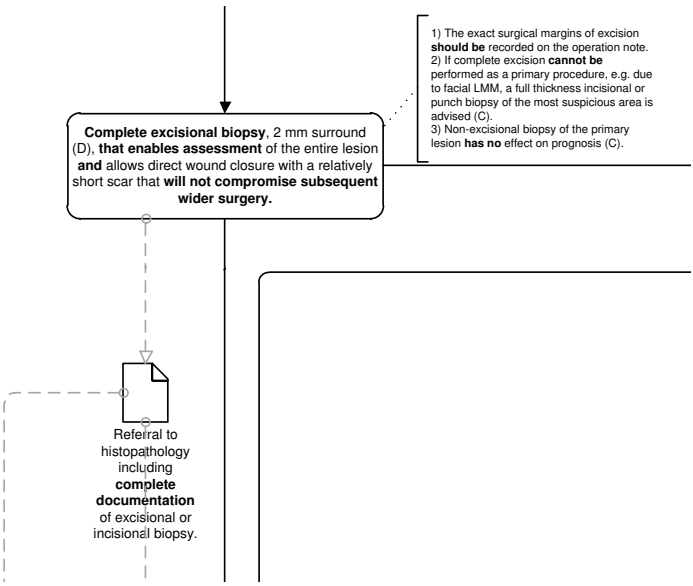


Figure 5: Referral after excision.

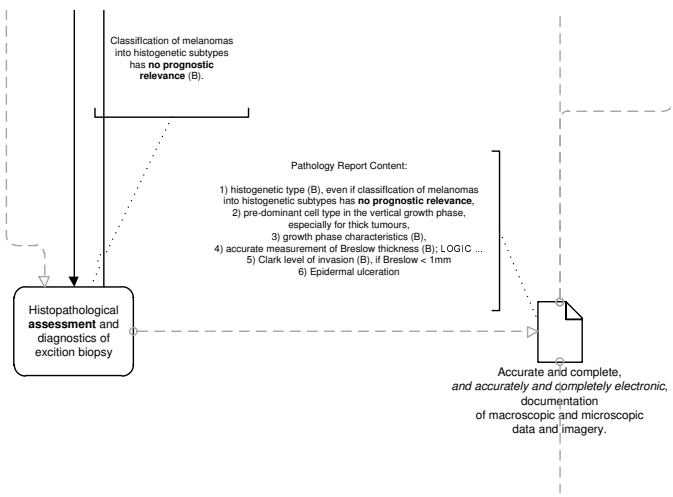


Figure 6: Pathologist histology examination.

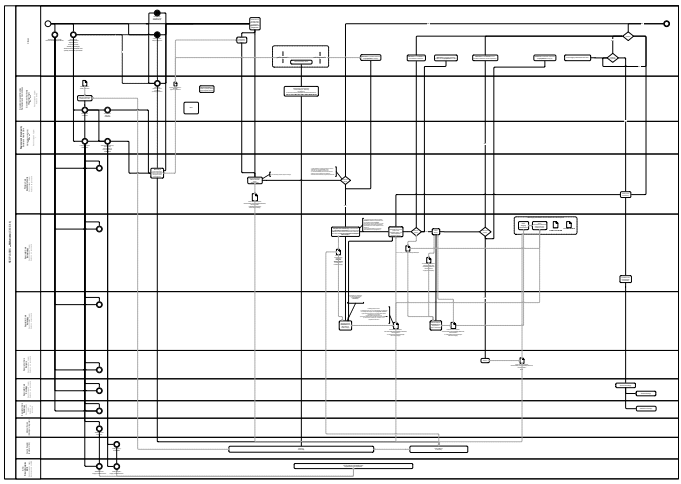


Figure 7: Overall BPMN encoded *NEOPLASMS – Melanoma (C43-C44)* process.

management like public authorities and ICT system suppliers are not covered at all.

The overall BPMN process[2] for "NEOPLASMS – Melanoma (C43-C44)" is given in Figure 7.

Several parts of this overall process are interesting, but in order to illuminate our logic approach, we will pick out only a few information and logical structure intensive parts.

Complete excisional biopsy is mostly performed by a dermatologist, but sometimes done within primary care by a general practitioner with special interest (GPwSI). This is sent over to a pathologist together with some form of a *referral to pathology report*. This process is shown in Figure 5.

As shown in Figure 6, the pathologist performs the histology, and sends back a *pathology report*, which then, together with the referral, enters the *clinical diagnosis and reassessment for local lymph node lymphadenopathy*. The outcome is Data Object

$$\texttt{DataObject}_{\texttt{ClinicalDiagnosisReport}}$$

2 The original document size is A0, and the smallest font size is 6pt.

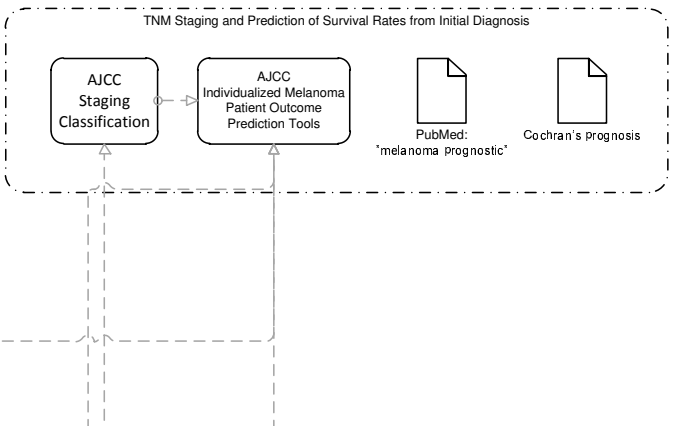


Figure 8: TNM staging and prediction as provided by AJCC.

representing diagnosis and reassessment as a basis for deciding about therapeutic lymph node dissection by *Fine Needle Aspiration Cytology* (FNAC). Lymph node involvement indicates an advanced stage of cancer, and metastatis including lymph nodes is an indication for radical dissection of that lymph node basin. The number of involved nodes is important, and it has been shown that ten year survival varies in the range of 20–45% as dependent on the number and extent of nodal involvement [9, 30].

Diagnosis encoding together with some specific typing has provided in the AJCC Cancer Staging Manual [38] and is included in the BPMN process diagram as shown in Figure 8. SNOMED CT terminologies can be included into sorts and operators, so that, e. g., "Melanoma in situ of face" as a constant of type `disorder` would be included. However, SNOMED CT for melanoma contains mainly disorder terms, which then overlap with diagnosis encoding, and they have therefore still not been included, since it only leads to mapping requirements for diagnosis and disorder codes and thus not add anything essential into other parts of the melanoma logic. Observable entities, findings, morphologic abnormalities, and procedures do exist, like "Clark's melanoma level" and "Breslow depth staging for melanoma" of type `observable_entity`, "Clark melanoma level ..." of type `finding`,

and "Excision of melanoma" of type `procedure`, but the scope is incomplete, e. g., as compared with terminology found in the AJCC Cancer Staging Manual and melanoma clinical guidelines.

6.3 BPMN

BPMN process design aims at being closer to human workflow, whereas process implementation (using BPEL) in closer to computer language. BPMN comes with syntax but possess no semantics. BPMN enables not just process information but in particular the communication of process information between participants in the process. Indeed, nothing is said about semantics, and therefore clearly nothing is specified concerning logic and the way participants act and infer with and by tasks and sub-process in the process as a whole.

BPMN diagrams build syntactically upon four basic categories of elements, namely Flow Objects, Connecting Objects, Artifacts and Swimlanes. Flow Objects, represented by Events, Activities and Gateways, define the behaviour of processes. Start and End are typical Event elements. Task and Sub-Process are the most common Activities. There are three Connecting Objects, namely

1. Sequence Flow,
2. Message Flow, and
3. Association.

Gateways, as Event elements, handle branching, forking, merging, and joining of paths. Pragmatically it is easy to see most of the logic resides in gateways, but this logic is external to BPMN in similar way as logic is external to imperative languages.

A Data Object is an Artifact, and having no effect on Sequence Flow or Message Flow. Data Objects are indeed seen to "represent" data, even if BPMN does not at all specify these representation formats or rules for such representations. However, Data Objects are expected to *provide information about what activities require to be performed and/or what they produce* [113, p. 93]. Information produced is in our sense the result of a reduction or inference, with related substitutions.

BPMN indeed provides no semantics but rather gives some hints where semantic issues eventually will become visible. A Sequence Flow is expected to *show the order that activities will be performed in a Process*, but nothing is said about overlaps. A Message Flow is expected to *show the flow of messages between two participants that are prepared to send and receive them*. Synchrony or asynchrony is – intentionally – not mentioned at this point.

6.4 BPMN DATA OBJECT INFORMATION SEMANTICS

As previously mentioned, the notion of a "dermatologist's *referral* to histology" or "pathology *report*" in terms of their content and data formats is well understood but this is not the case when considering the referrals and reports, as *documents*, as a whole. To better understand the documents as a whole we must consider in detail the notions of documents, document structures, and document templates. In the categorical framework outlined in previous sections we would identify *a document over a signature* Σ with the notion of a ground term over Σ, i. e., a term containing no variables. The set $\mathsf{T}_\Sigma(\emptyset)$ then corresponds to the set of all documents over the signature Σ. In this interpretation a *document template over a signature* Σ then is a non-ground Σ-term over some set of variables X. That is, $\mathsf{T}_\Sigma(X) \setminus \mathsf{T}_\Sigma(\emptyset)$ is the set of all document templates. We may underline here that the report structure really *is* the signature Σ?

Our suggestion for information semantics in BPMN is then that Data Object coincides with document and, by extension, is a ground term. For documentations and document refinements, this means in reality that we call a document draft a 'document template' all the way until it has been matured to become the "finished" document, where all variables have been instantiated with relevant information (ground terms). Similarly, 'token' coincides with variable substitution. From this view of BPMN information semantics, we are able to extract at any point in the data flow a valid variable substitution that precisely represents an information snapshot of the process at that particular point. An activity in the BPMN sense can then be viewed as a composition of vari-

able substitutions with the initial token or variable substitution being the Kleisli category identity morphism $\eta\colon X \longrightarrow \mathsf{T}_\Sigma X$.

In order to provide examples, let us briefly outline how the underlying signatures as 'owned' and recognized by respective disciplines might look like. In our example we may start with the signatures,

$$\Sigma_{\text{dermatology}} = (S_{\text{dermatology}}, \Omega_{\text{dermatology}}) \text{ and}$$
$$\Sigma_{\text{pathology}} = (S_{\text{pathology}}, \Omega_{\text{pathology}}),$$

respectively, for the professional groups as represented by dermatologists and pathologists. An overall signature

$$\Sigma_{\text{DataObject}} = (S_{\text{DataObject}}, \Omega_{\text{DataObject}}),$$

not specified in detail at this point, would then embrace the constituent signatures $\Sigma_{\text{dermatology}}$ and $\Sigma_{\text{pathology}}$, so that $S_{\text{dermatology}}$ and $S_{\text{pathology}}$, respectively, are suitably embedded into $S_{\text{DataObject}}$, and similarly $\Omega_{\text{dermatology}}$ and $\Omega_{\text{pathology}}$, respectively, are suitably embedded into $\Omega_{\text{DataObject}}$.

When combining sets of sorts and operators, respectively, we must use some useful categorical limits so as to, for example, avoid naming conflicts. This is comparable to the situation of modules in programming languages, the same symbol may be defined in multiple modules and when combining these modules we must be careful to denote which of these definitions we wish to use.

We may aim at providing clinical diagnosis reports as terms `t` being of sort $\texttt{ClinicalDiagnosisReport} \in S_{\text{dermatology}}$. This may also be denoted by `t` :: `ClinicalDiagnosisReport`. This term `t` is then seen as produced by a number of operators, manipulating and attaching terms in form of various 'subdocuments', like a histology report to be integrated with the clinical diagnosis report. Assume then we have the histology report, as a term being

`u` :: `HistologyReport`,

with $\texttt{HistologyReport} \in S_{\text{pathology}}$. The term `u` is then typically delivered by the pathologist as a response to a referral,

`v` :: `ReferralToPathologyReport`,

with $\texttt{ReferralToPathologyReport} \in S_{\text{dermatology}}$, from the dermatologist after excision of the mole.

The refinement of clinical reports are assuming evidence-based knowledge as typically appearing in guidelines of various kind and with different level of detail.

As an example, the *Scottish Intercollegiate Guidelines Network* (SIGN), maintains clinical guidelines in different areas targeted for various professional groups. A sort representing these guidelines could then be $\texttt{SIGN} \in S_{\text{DataObject}}$, with the particular guideline for *Cutaneous Melanoma* [127] being a document as a term $t_{\text{SIGN}_{72}} :: \texttt{SIGN}$.

Report production, enhancement and enrichment is then enabled, e. g., by operators like $\texttt{RefineReport}_{\texttt{s}} : \texttt{s} \to \texttt{s}$, where in the case of $\texttt{s} \in S_{\text{dermatology}}$ it is a report production authored by a dermatology clinic. Clinical diagnosis and reassessment following SIGN guidelines can then be seen, on the one hand hand, as a documented by the operator

$$\texttt{ClinicalDiagnosisAndReassessment}_{\texttt{s}} : \texttt{SIGN} \times \texttt{s} \to \texttt{s},$$

with $\texttt{SIGN} \in S_{\text{DataObject}}$ and $\texttt{s}$ being members of the same set of sorts attached to the particular discipline. On the other hand, making the origin of the histology report more explicit,

$$\texttt{ClinicalDiagnosisAndReassessment}$$

could have as input terms typed within $S_{\text{pathology}}$, $S_{\text{dermatology}}$, and $S_{\text{DataObject}}$ in general, and having a type in $S_{\text{pathology}}$ as output type. In this case, the operator goes "across" $\Sigma_{\text{dermatology}}$ and $\Sigma_{\text{pathology}}$, so that the underlying signature is more than just something corresponding to a coproduct of signatures.

Some more specific typing can also be mentioned, such as the sort for WHO's *International Classification of Diseases* $\texttt{ICD}-\texttt{10} \in S_{\text{DataObject}}$, e. g., with the constant $\texttt{C43.5} : \to \texttt{ICD}-\texttt{10}$ representing *malignant melanoma of trunk*. A parallel classification related to dermatology related diseases exists by the *International League of Dermatological Societies* (ILDS), and this could then be typed by $\texttt{ILDS} \in S_{\text{dermatology}}$, where the $\texttt{C43.Lx0} : \to \texttt{ILDS}$ diagnosis codes

do not have exact counterparts in ICD-10. The type of melanoma could be encoded by $\texttt{TypeOfMelanoma} \in S_{\text{dermatology}}$ with

$$\texttt{SSM}, \texttt{NM}, \texttt{LMM}, \texttt{ALM}: \ \to \texttt{TypeOfMelanoma}$$

representing *superficial spreading melanoma* (SSM), *nodular melanoma* (NM), *lentigo maligna melanoma* (LMM) and *acral lentiginous melanoma* (ALM), respectively. Further, *TNM staging* can be similarly encoded with

$$\texttt{T}, \texttt{N}, \texttt{M}, \texttt{Stage} \in S_{\text{cancer\_staging}},$$

with

$$\begin{aligned} \texttt{T0},\ldots: &\ \to \texttt{T}, \\ \texttt{N0},\ldots: &\ \to \texttt{N}, \\ \texttt{M0},\ldots: &\ \to \texttt{M}, \text{and} \\ \texttt{ERROR}, \texttt{UNK}, \texttt{0}, \texttt{IA}, \texttt{IB}, \texttt{IIA}, \ldots, \texttt{IIIC}, \texttt{IIINOS}, \texttt{IV}: &\ \to \texttt{Stage}. \end{aligned}$$

The *AJCC TNM 7 Staging* is then an operator

$$\texttt{AJCC\_TNM\_7\_Stage}: \texttt{T} \times \texttt{N} \times \texttt{M} \to \texttt{Stage},$$

whereas the staging 'reconstructed' in the SIGN guideline for cutaneous melanoma is $\texttt{SIGN\_TNM\_Stage}: \texttt{T} \times \texttt{N} \times \texttt{M} \to \texttt{Stage}$. The semantics (or algebras) for these two should be the same but the table formats in SIGN and AJCC look different, so the equivalence must really be 'constructed' as a mapping between those tables. *Clark levels* and *Breslow thickness*, as appearing in histology reports, can be encoded similarly, and these values are related, e. g., by the fact that Clark level of invasion yields additional prognostic information in patients with a Breslow thickness of $<$1 mm [27].

Usually, second opinions are not requested, but theoretically we could imagine two independent pathologists providing a histology report related precisely to the same mole. How are then these two reports,

$$\texttt{t}_{\texttt{HistologyReport}_1}, \texttt{t}_{\texttt{HistologyReport}_2} :: \texttt{HistologyReport},$$

handled and integrated into the clinical reports so that some kind of "confirmation effect" is allowed to take place in that integration of histological results and observation. We will return to this issue when discussing semantics of tokens and token algebras.

As an example, consider a clinical report. We may have as a template of such a report a non-ground term

$$\texttt{ClinicalDiagnosisAndReassessment}(\ldots, x_{\texttt{HistologyReport}}, \ldots)$$

with $x_{\texttt{HistologyReport}}$ being a variable of sort `HistologyReport`. The particular activity performed by the pathologist (having a unique personal ID, say 12345) will then give rise to a many-sorted substitution σ such that

$$\sigma_s(x) = \begin{cases} t & \text{if } s = \texttt{HistologyReport} \\ x & \text{otherwise} \end{cases}$$

where

$$\begin{aligned} t = \texttt{HistologyReport}(&\texttt{PathId}(12345), \\ &\texttt{BreslowThickness}(<1\,\text{mm}), \\ &\texttt{ClarkeLevel}(\ldots), \\ &\ldots). \end{aligned}$$

The term resulting from this substitution – which may still be a template, but nevertheless is closer to a document – will be

$$\texttt{ClinicalReport}(\ldots, \texttt{HistologyReport}(\texttt{PathId}(12345), \ldots), \ldots)$$

and we can of course present the whole report in a more attractive form, e. g., as shown in Figure 9.

Note, this view of BPMN information semantics differ from Eklund et al. [52] where Data Objects were taken to be variable substitutions. Also note, that BPMN tokens are seen, e. g., not to traverse BPMN Message Flows, which means that tokens are not explicitely related with BPMN Data Objects in our semantic interpretation of Data Objects. Generally, the intuition of BPMN tokens resembles that of tokens use in Petri nets, which supports our view that tokens carry substitutions, since data is managed in the executions that provide required state changes.

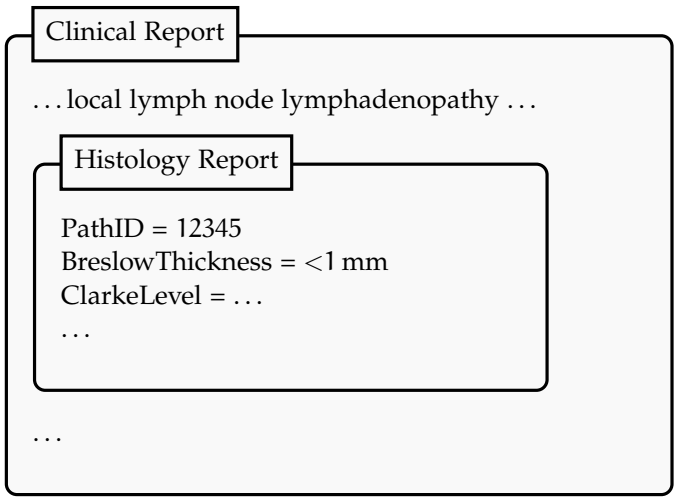


Figure 9: Alternative, more attractive presentation of a clinical report term.

6.5 BPMN TOKEN SEMANTICS

Tokens will be seen as semantically represented by substitutions. We also aim at an "algebra of substitutions", in the sense of having operators that combine certain existing substitutions into richer substitutions, typically either enriching by composing substitutions so as to provide a higher level of completion of particular documents, or reducing uncertainties attached with some facts residing in reports, e. g., by providing "second opinions", and having the possibility to blend various opinions into one. These operators can be used to produce similar operators for providing richness in documents, even if we will not have a corresponding "algebra of documents".

Kleene algebras are suitable structures for these algebras of substitutions, and key to these solutions, in particular in order to define the Kleene asterate, is the use of partially ordered monads, and the way the compose with the many-sorted term monad providing new partially ordered monads.

In Chapter 2 we saw that $\mathbf{L} \bullet \mathbf{T}_\Sigma$, with $\mathbf{T}_\Sigma$ being a one-sorted term monad and $\mathbf{L}$ as the many-valued powerset monad, can be extended to a partially ordered monad. We further showed that $(\mathrm{Hom}(X, \mathsf{L}\mathsf{T}_\Sigma X), +, \cdot, ^*, 0, 1)$, i. e., the set of morphisms in the Kleisli category $\mathtt{Set}_{\mathbf{L}\bullet\mathbf{T}_\Sigma}$, is a Kleene algebra.

Recall, we have $0 = 0_X$, i. e., $0_X(x) = 0$ for all $x \in X$, and $1 = \eta_X^{\mathbf{L}\bullet\mathbf{T}_\Sigma}$. Further, for substitutions $\sigma_1, \sigma_2 \in \mathrm{Hom}(X, \mathsf{L}\mathsf{T}_\Sigma X)$, we have

$$\sigma_1 + \sigma_2 = \sigma_1 \vee \sigma_2,$$

i. e., pointwise according to $(\sigma_1 + \sigma_2)(x) = \sigma_1(x) \vee \sigma_2(x)$, and

$$\sigma_1 \cdot \sigma_2 = \sigma_1 \diamond \sigma_2$$

where $\sigma_1 \diamond \sigma_2 = \mu_X^{\mathbf{L}\bullet\mathbf{T}} \circ \mathsf{L}\mathsf{T}\sigma_2 \circ \sigma_1$ is the composition of morphisms in the corresponding Kleisli category of $\mathbf{L} \bullet \mathbf{T}$.

Let us now consider the many-sorted case. If we assume for a family of variables X_S that all constituent X_s, $s \in S$ are disjoint, i. e., the variable name uniquely determines the variable sort then this Kleene monad situation in the one-sorted case can be extended to the many-sorted case since a many-sorted term monad $\mathbf{T}_\Sigma$ can operate over $\mathtt{Set}$ directly. We will, however, retain the convenient notation $(X_s)_{s\in S}$.

It is now possible to move forward towards a "partial algebra of documents". To illuminate this possibility, let

$$\mathsf{t}_{\mathtt{HistologyReport}} :: \mathtt{HistologyReport}$$

be a template, or "document in progress", as part of an overall term

$$(\mathsf{t}_s^{\mathrm{ScopeOfReport}})_{s\in S_{\mathrm{DataObject}}} \in \mathsf{T}_{\Sigma_{\mathrm{DataObject}}}(X_s)_{s\in S_{\mathrm{DataObject}}},$$

and let

$$\sigma_i \colon (X_s)_{s\in S_{\mathrm{DataObject}}} \longrightarrow \mathsf{T}_{\Sigma_{\mathrm{DataObject}}}(X_s)_{s\in S_{\mathrm{DataObject}}}$$

for $i = 1, 2$ be substitutions. Then

$$\mu \circ \mathsf{T}(\sigma_1 + \sigma_2)((\mathsf{t}_s^{\mathrm{ScopeOfReport}})_{s\in S_{\mathrm{DataObject}}})$$

is more of a 'concatenation', or composition of information along a path of tasks, of

$$\mu \circ \mathsf{T}(\sigma_1)((t_s^{\text{ScopeOfReport}})_{s \in S_{\text{DataObject}}}) \text{ and}$$
$$\mu \circ \mathsf{T}(\sigma_2)((t_s^{\text{ScopeOfReport}})_{s \in S_{\text{DataObject}}})$$

whereas

$$\mu \circ \mathsf{T}(\sigma_1 \cdot \sigma_2)((t_s^{\text{ScopeOfReport}})_{s \in S_{\text{DataObject}}})$$

is a corresponding *sharpening the uncertainties*, or enhancing truth values residing in that report. This can also be seen as providing and allowing for "second opinions".

The interaction between Data Objects and tokens is thereby clear, as token in form of substitutions provide documents in form of terms with required content. Tokens as information carriers traverse the process, possibly in parallel with other tokens, so a variable substitution applied for a term is like taking an information snapshop of the process at that particular point.

7 CONCLUSIONS

We have in this thesis considered the question of non-classical logics and information representation from the perspective, firstly, of terms arising from strictly categorically constructed term monads over possibly enriched notions of signatures and, secondly, in the framework of generalized general logics, a generalization of the well known general logics framework introduced by Meseguer [105]. For this generalization we provided a specialization to logics having an underlying notion of terms in a general sense. Indeed, throughout all constructions and considerations given we have had a focus, implicit and explicit, on terms and their use in richer information representations for logic. This text may therefore be seen as a counter point to the strong focus given by the fuzzy logics community to manipulation of truth values. Indeed, we wish to make the point that a comprehensive view of non-classical logic must not limit itself to just non-classical notions of truth values but must also consider the other aspects of logic, of which we here have focused on non-classical notions of terms, term sets, and sets of sentences. Of course, this is not to say that non-classical notions of truth is unimportant but rather that other notions are not unimportant.

All this rely on a sufficiently strict definition of the term constructions. This is often surprisingly difficult and, in particular, the transition from the one-sorted case to the many-sorted case is usually underestimated, as the one-sorted case allows for viewing subobjects mostly as subsets. Shifting to the many-sorted case reveals subtleties that are not identified without a rigorous and formal treatment of the underlying structures of objects and morphisms, i. e., category theory as a meta language is most suitable in these respects. We have clearly shown how the many-sorted term monad is not simply a component-wise assembly of one-sorted functors and monads, which makes distributive laws and management of underlying categories non-trivial. As shown

with the many-valued powerset monad, however, distributive laws are possible even in the many-sorted case. Other possibilities remain to show formally, e. g., to establish a distributive law for the Giry monad [63] providing probabilistic notions.

For the theoretical constructions we discussed a few examples within the field of computer science, primarily concerning type theoretical aspects and their corresponding developments in description logic and social choice theory. We also considered the importance of partially ordered monads and their extension to Kleene monads. These latter developments, in particular showing that Kleene monads give rise to Kleene algebras in a intuitive manner clearly invites to further developments in program semantics and program verification. Such developments may, e. g., involve possible generalizations of the work of Kozen [92] on the semantics of `while` loops and `if` statements.

Applications outside the field of computer science and mathematics that involve all the subtleties for uncertainty management are important and we find health and social care as an application area that has always been aware of needs to be more precise about terminologies. In recent years, formal logic is being recognized as valuable for providing enrichments to, e. g., ontology. Processes and workflow have also been in focus over decades, but languages for processes have generally not been of much concern. This is seen among several national authorities for social and health care, and in their on-going efforts to provide various canonic and generic process views. To this end we considered one possible approach using the framework provided by BPMN applied in the term monad setting.

BPMN is being used more and more, even if motivations for using BPMN has not been expressed thoroughly. Technically, BPEL adds value to BPMN developments, in particular in BPMN to BPEL transformations are considered at least to some extent during the course of BPMN developments. We have shown how terms and document templates can be seen as commonly embraced by several or even all Swimlanes in a process. Moving forward from terms to sentences, and more broadly to cover all aspects and building blocks of logic, shows how logic on the other hand resides more within respective Swimlanes. However, as

these logics can be seen to constitute a *category of substitution logics* [46, 47], creating overall logics across Swimlanes is in principe possible. Mappings between logics, or morphism between objects in the category of substitution logics, are then typically mappings between logics residing in respective disciplines, or mappings between logics residing in an authority and a discipline.

Future work in these respects is further enhancing the melanoma care process, e. g., involving details for other disciplines as Swimlanes in the process, and providing more detail on underlying signatures as a basis for logic enhancements followed by specific rule base development. Guidelines are typical examples that should not be seen only as terms, or documents, but indeed contain evidence-based rules, that need to be organized more logically as they emerge mostly from statistical machineries. 'Evidence' in the medical domain is really not 'logical evidence', but 'statistical evidence' as support for decision-making, and indeed as provided by statistics guided observations and analysis. Moving over from statistics to logic is quite a giant paradigm leap, and as statistics basically come with no logic structure, these logical structures have to identified. Identifying sentences and proof calculi must then not bypass initial productions of underlying signature, i. e., rigorous typing of the underlying information and ontologies, and in our example application and the monograph as a whole we have aimed precisely at demonstrating this situation.

Clearly then it is the case that a great deal of effort lies ahead. Both in further theoretical developments and in further applications. The full potential of non-classical notions of terms and substitution logics still is in many ways undeveloped and this thesis barely scratches the surface of the iceberg.

BIBLIOGRAPHY

[1] Samson Abramsky and Achim Jung. Domain theory. In S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume III, pages 1–168. Oxford University Press, 1994. ISBN 0-19-853762-X.

[2] Jiří Adámek, Horst Herrlich, and George Strecker. *Abstract and concrete categories*. Wiley-Interscience, New York, NY, USA, 1990. ISBN 0-471-60922-6.

[3] Jaume Agustí-Cullell, Francesc Esteva, Pere Garcia, and Lluis Godo. Formalizing multiple-valued logics as institutions. In Bernadette Bouchon-Meunier, Ronald R. Yager, and Lotfi A. Zadeh, editors, *IPMU*, volume 521 of *Lecture Notes in Computer Science*, pages 269–278. Springer, 1990. ISBN 3-540-54346-5.

[4] Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, 1974. ISBN 0-201-00029-6.

[5] American Psychiatric Association. *Diagnostic and Statistical Manual of Mental Disorders DSM-IV-TR Fourth Edition (Text Revision)*. American Psychiatric Association, Washington, DC, 4th edition, July 2000.

[6] Kenneth J. Arrow. *Social choice and individual values*. Wiley, New York, 1951.

[7] Egidio Astesiano, Michel Bidoit, Hélène Kirchner, Bernd Krieg-Brückner, Peter D. Mosses, Donald Sannella, and Andrzej Tarlecki. CASL: the Common Algebraic Specification Language. *Theor. Comput. Sci.*, 286(2):153–196, 2002.

[8] Steve Awodey. *Category Theory*. Oxford Logic Guides. Oxford University Press, 2006. ISBN 978-0-19-856861-2.

[9] C.M. Balch, S.J. Soong, J.E. Gershenwald, J.F. Thompson, D.S. Reintgen, N. Cascinelli, M. Urist, K.M. McMasters, M.I. Ross, J.M. Kirkwood, et al. Prognostic factors analysis of 17,600 melanoma patients: validation of the american joint committee on cancer melanoma staging system. *Journal of Clinical Oncology*, 19(16):3622, 2001.

[10] Stefan Banach. *Théorie des opérations linéares*. Warsaw, 1932.

[11] Taradas Bandyopadhyay. Choice procedures and rational selections. *Annals of Operations Research*, 80:49–66, 1980.

[12] Taradas Bandyopadhyay. Choice procedures and power structure in social decisions. *Social Choice and Welfare*, 37 (4):597–608, 2011.

[13] Henk P. Barendregt. Introduction to lambda calculus. *Nieuw Archief voor Wisenkunde*, 4(2):337–372, 1984.

[14] Michael Barr and Charles Wells. *Category theory for computing science*. Prentice-Hall, Inc., 1990. ISBN 0-13-120486-6.

[15] Jon Beck. Distributive laws. In *Seminar on Triples and Categorical Homology Theory, 1966/67, Lecture Notes in Mathematics* **80**, pages 119–140. Springer-Verlag, 1969.

[16] John L. Bell. Category theory and the foundations of mathematics. *British Journal for the Philosophy of Science*, 32(4): 349–358, December 1981.

[17] Jean Bénabou. Catégories avec multiplication. *C. R. Acad. Sci. Paris*, 256:1887–1890, 1963.

[18] Jean Bénabou. Algébre élémentaire dans les catégories avec multiplication. *C. R. Acad. Sci. Paris*, 258:771–774, 1964.

[19] Jean Bénabou. Structures algébriques dans les catégories. *Cahiers de Topologie et Géométrie Différentielle Catégoriques*, 10 (1):1–126, 1968.

[20] Garrett Birkhoff. On the structure of abstract algebras. *Mathematical Proceedings of the Cambridge Philosophical Society*, 31:433–454, 1935. ISSN 1469-8064. doi: 10.1017/S0305004100013463.

[21] Garrett Birkhoff. Generalized arithmetic. *Duke Math. J.*, 9 (2):283–302, 1942.

[22] Garrett Birkhoff. *Lattice Theory*, volume 25. American Mathematical Society, third edition, 1995.

[23] Ronald J. Brachman and Hector J. Levesque. The tractability of subsumption in frame-based description languages. In Ronald J. Brachman, editor, *AAAI*, pages 34–37. AAAI Press, 1984. ISBN 0-262-51053-7.

[24] Ronald J. Brachman and James G. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171–216, 1985.

[25] Stanley Burris and Hanamantagouda P. Sankappanavar. *A Course in Universal Algebra*. Springer-Verlag, 1981.

[26] Rod M. Burstall and Joseph A. Goguen. The semantics of CLEAR, a specification language. In Dines Bjørner, editor, *Abstract Software Specifications*, volume 86 of *Lecture Notes in Computer Science*, pages 292–332. Springer, 1979. ISBN 3-540-10007-5.

[27] P. Büttner, C. Garbe, J. Bertz, G. Burg, B. D'Hoedt, H. Drepper, I. Guggenmoos-Holzmann, W. Lechner, A. Lippold, C.E. Orfanos, et al. Primary cutaneous melanoma. optimized cutoff points of tumor thickness and importance of clark's level for prognostic classification. *Cancer*, 75(10): 2499–2506, 1995.

[28] Alonzo Church. A formulation of the simple theory of types. *The journal of symbolic logic*, 5(2):56–68, 1940.

[29] A. J. Cochran, D. Elashoff, D. L. Morton, and R. Elashoff. Individualized prognosis for melanoma patients. *Human Pathology*, 31(3):327–331, 2000.

[30] D.G. Coit, A. Rogatko, and M.F. Brennan. Prognostic factors in patients with melanoma metastatic to axillary or inguinal lymph nodes. a multivariate analysis. *Annals of surgery*, 214(5):627, 1991.

[31] Nicolas Condorcet. *Essai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix*. Imprimerie royale, Paris, France, 1785.

[32] Haskell B. Curry. Grundlagen der kombinatorischen logik. *American journal of mathematics*, 52(4):789–834, 1930.

[33] Jean C. de Borda. *Memoire sur les Elections au Scrutin*. Histoire de l'Academie Royale des Sciences, Paris, 1781.

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

[35] Răzvan Diaconescu. *Three decades of institution theory*, pages 309–322. Studies in Universal Logic. Springer Basel, 2012. ISBN 9783034601443.

[36] Răzvan Diaconescu. Institutional semantics for many-valued logics. *Fuzzy Sets and Systems*, 218(0):32–52, 2013. ISSN 0165-0114. doi: 10.1016/j.fss.2012.11.015.

[37] Anton Dumitriu. *History of Logic*, volume I. Abacus Press, 1977.

[38] S.B. Edge, D.R. Byrd, C.C. Compton, A.G. Fritz, FL Greene, and A. Trotti. *AJCC Cancer Staging Manual*. Springer, 7th edition, 2010.

[39] S. Eilenberg, D. K. Harrison, S. MacLane, and H. Röhrl, editors. *Proceedings of the Conference on Categorical Algebra, La Jolla 1965*, Berlin, 1966. Springer-Verlag. ISBN 978-3-642-99904-8.

[40] Samuel Eilenberg and G. Max Kelly. Closed categories. In Eilenberg et al. [39], pages 421–562. ISBN 978-3-642-99904-8. doi: 10.1007/978-3-642-99902-4_22.

[41] Samuel Eilenberg and Saunders MacLane. General theory of natural equivalences. *Transactions of the American Mathematical Society*, 58(2):231–294, September 1945. doi: 10.2307/1990284.

[42] Samuel Eilenberg and John C. Moore. Adjoint functors and triples. *Illinois Journal of Mathematics*, 9(3):381–398, 1965.

[43] Patrik Eklund. Signatures for assessment, diagnosis and decision-making in ageing. In Hüllermeier et al. [80], pages 271–279. ISBN 978-3-642-14057-0. doi: 10.1007/978-3-642-14058-7_27.

[44] Patrik Eklund and Werner Gähler. Fuzzy filter functors and convergence. In Stephen Ernest Rodabaugh, Erich Peter Klement, and Ulrich Höhle, editors, *Applications of category theory to fuzzy subsets*, pages 109–136. Kluwer Academic Publishers, 1992.

[45] Patrik Eklund and Werner Gähler. Completions and compactifications by means of monads. In R. Lowen and M. Roubens, editors, *Fuzzy Logic*, pages 39–56. Kluwer Academic Publishers, Dordrecht, 1993.

[46] Patrik Eklund and Robert Helgesson. Monadic extensions of institutions. *Fuzzy Sets and Systems*, 161(18):2354–2368, 2010. ISSN 0165-0114. doi: 10.1016/j.fss.2010.03.002. A tribute to Ulrich Höhle on the occasion of his 65th birthday.

[47] Patrik Eklund and Robert Helgesson. Information ontology and process structure in social and health care. 2012.

[48] Patrik Eklund, M.Ángeles Galán, Manuel Ojeda-Aciego, and Agustin Valverde. Set functors and generalised terms. *Proc. IPMU 2000, 8th Information Processing and Management of Uncertainty in Knowledge-Based Systems Conference*, III:1595–1599, 2000.

[49] Patrik Eklund, Agnieszka Rusinowska, and Harrie C. M. de Swart. Consensus reaching in committees. *European Journal of Operational Research*, 178(1):185–193, 2007.

[50] Patrik Eklund, Robert Helgesson, and Helena Lindgren. Towards refinement of clinical evidence using general logics. In Leszek Rutkowski, Ryszard Tadeusiewicz, Lotfi A. Zadeh, and Jacek M. Zurada, editors, *ICAISC*, volume 5097 of *Lecture Notes in Computer Science*, pages 1029–1040. Springer, 2008. ISBN 978-3-540-69572-1.

[51] Patrik Eklund, Agnieszka Rusinowska, and Harrie C. M. de Swart. A consensus model of political decision-making. *Annals of Operations Research*, 158(1):5–20, 2008.

[52] Patrik Eklund, Mats Johansson, Johan Karlsson, and Rickard Åström. BPMN and its semantics for information management in emergency care. In *Proceedings of the 2009 Fourth International Conference on Computer Sciences and Convergence Information Technology*, ICCIT '09, pages 273–278, Washington, DC, USA, 2009. IEEE Computer Society. ISBN 978-0-7695-3896-9. doi: 10.1109/ICCIT.2009.27.

[53] Patrik Eklund, Jari Kortelainen, and Lawrence Neff Stout. Introducing fuzziness in monads: cases of the term monad and the powerobject monad. In *Proc. LINZ2009, 30th Linz Seminar on Fuzzy Set Theory, Linz, Austria*, 2009.

[54] Patrik Eklund, Mario Fedrizzi, and Hannu Nurmi. A categorical approach to the extension of social choice functions. In Hüllermeier et al. [80], pages 261–270. ISBN 978-3-642-14057-0. doi: 10.1007/978-3-642-14058-7.

[55] Patrik Eklund, M.Ángeles Galán, Robert Helgesson, and Jari Kortelainen. Paradigms for many-sorted non-classical substitutions. In Jaakko Astola and Radomir S. Stankovic, editors, *ISMVL*, pages 318–321. IEEE, 2011. ISBN 978-0-7695-4405-2. doi: 10.1109/ISMVL.2011.10.

[56] Patrik Eklund, Jari Kortelainen, and Lawrence Neff Stout. Adding fuzziness to terms and powerobjects using a monadic approach. *Fuzzy Sets and Systems*, 192:104–122, 2012.

[57] Patrik Eklund, M.Ángeles Galán, Robert Helgesson, and Jari Kortelainen. Fuzzy terms. *Fuzzy Sets and Systems*, (0):–, 2013. ISSN 0165-0114. doi: 10.1016/j.fss.2013.02.012.

[58] Kfir Eliaz. Social aggregators. *Social Choice and Welfare*, 22: 317–330, 2004. ISSN 0176-1714.

[59] Werner Gähler. Monads and convergence. In *Generalized Functions, Convergences Structures, and Their Applications*, pages 29–46, New York, 1988. Plenum Press.

[60] Werner Gähler. General topology – the monadic case, examples, applications. *Acta Mathematica Hungarica*, 88(4):279–290, September 2000.

[61] Werner Gähler. Extension structures and completions in topology and algebra. Seminarberichte aus dem Fachbereich Mathematik Band 70, FernUniversität Hagen, 2001.

[62] Werner Gähler and Patrik Eklund. Extension structures and compactifications. In *Categorical Methods in Algebra and Topology*, pages 181–205, 2000.

[63] Michèle Giry. A categorical approach to probability theory. In B. Banaschewski, editor, *Categorical Aspects of Topology and Analysis*, volume 915 of *Lecture Notes in Mathematics*, pages 68–85. Springer Berlin Heidelberg, 1982. ISBN 978-3-540-11211-2. doi: 10.1007/BFb0092872.

[64] Roger Godement. *Topologie algébrique et théorie des faisceaux*. Hermann, Paris, 1958.

[65] Lluis Godo, Francesc Esteva, Pere García, and Jaume Agustí-Cullell. A formal semantical approach to fuzzy logic. In *ISMVL*, pages 72–79, 1991.

[66] Joseph A. Goguen. L-fuzzy sets. *Journal of Mathematical Analysis and Applications*, 18(1):145–174, 1967. ISSN 0022-247X. doi: 10.1016/0022-247X(67)90189-8.

[67] Joseph A. Goguen. What is unification? A categorical view of substitution, equation and solution. In Maurice Nivat

and Hassan Aït-Kaci, editors, *Resolution of Equations in Algebraic Structures, Volume 1: Algebraic Techniques*, pages 217–261. Academic Press, 1989.

[68] Joseph A. Goguen. A categorical manifesto. *Mathematical Structures in Computer Science*, 1(1):49–67, March 1991.

[69] Joseph A. Goguen. Three perspectives on information integration. In Yannis Kalfoglou, W. Marco Schorlemmer, Amit P. Sheth, Steffen Staab, and Michael Uschold, editors, *Semantic Interoperability and Integration*, volume 04391 of *Dagstuhl Seminar Proceedings*. IBFI, Schloss Dagstuhl, Germany, 2005.

[70] Joseph A. Goguen and Rod M. Burstall. Introducing institutions. In *Proceedings of the Carnegie Mellon Workshop on Logic of Programs*, pages 221–256, London, UK, 1984. Springer-Verlag. ISBN 3-540-12896-4.

[71] Joseph A. Goguen and Rod M. Burstall. INSTITUTIONS: Abstract model theory for specification and programming. *J. ACM*, 39(1):95–146, 1992.

[72] Joseph A. Goguen and José Meseguer. Completeness of many-sorted equational logic. *SIGPLAN Not.*, 17(1):9–17, 1982. ISSN 0362-1340. doi: 10.1145/947886.947887.

[73] Joseph A. Goguen and José Meseguer. Order-sorted algebra I: Equational deduction for multiple inheritance, overloading, exceptions and partial operations. *Theor. Comput. Sci.*, 105(2):217–273, 1992.

[74] Siegfried Gottwald. A treatise on many-valued logics. In *Studies in Logic and Computation*. Press, 2001.

[75] Carl A. Gunter. The semantics of types in programming languages. In S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science, Volume 3: Semantic Structures*, pages 395–475. Clarendon Press, Oxford, 1994.

[76] Petr Hájek. *The Metamathematics of Fuzzy Logic*. Kluwer, 1998.

[77] Robert Helgesson. A categorical approach to logics and logic homomorphisms. Master's thesis, Umeå University, 2007. UMNAD 676/07.

[78] Ulrich Höhle and Lawrence Neff Stout. Foundations of fuzzy sets. *Fuzzy Sets and Systems*, 40(2):257–296, April 1991. ISSN 0165-0114. doi: 10.1016/0165-0114(91)90163-K.

[79] Peter J. Huber. Homotopy theory in general categories. *Mathematische Annalen*, 144(5):361–385, 1961.

[80] Eyke Hüllermeier, Rudolf Kruse, and Frank Hoffmann, editors. *Information Processing and Management of Uncertainty in Knowledge-Based Systems. Applications - 13th International Conference, IPMU 2010, Dortmund, Germany, June 28 - July 2, 2010. Proceedings, Part II*, volume 81 of *Communications in Computer and Information Science*, 2010. Springer. ISBN 978-3-642-14057-0. doi: 10.1007/978-3-642-14058-7.

[81] Kiyosi Itô. *Encyclopedic Dictionary of Mathematics*. Number 2. MIT Press, 1993. ISBN 9780262590204.

[82] André Joyal and Ross Street. Braided tensor categories. *Advances in Mathematics*, 102(1):20–78, 1993. ISSN 0001-8708. doi: 10.1006/aima.1993.1055.

[83] Daniel M. Kan. Adjoint functors. *Transactions of the American Mathematical Society*, 87(2):294–329, 1958.

[84] Hans Keiding. The categorical approach to social choice theory. *Mathematical Social Sciences*, 1(2):177–191, 1981. ISSN 0165-4896. doi: 10.1016/0165-4896(81)90005-6.

[85] H. H. Keller. Die limes-uniformisierbarkeit der limesräume. *Mathematische Annalen*, 176(4):334–341, 1968. ISSN 0025-5831. doi: 10.1007/BF02052894.

[86] John L. Kelley. *General Topology*. Van Nostrand, 1955.

[87] G. Max Kelly. On Mac Lane's conditions for coherence of natural associativities, commutativities, etc. *J. Algebra*, 1: 397–402, 1964.

[88] G. Max Kelly. *Basic concepts of enriched category theory*. Number 64 in London Mathematical Society Lecture Notes. Cambridge University Press, 1982. ISBN 0521287022.

[89] Claude Kirchner and Hélène Kirchner. *Rewriting Solving Proving*. Self-published, January 2006. Preliminary version.

[90] Stephen C. Kleene. Representation of events in nerve nets and finite automata. In C. Shannon and J. Mccarthy, editors, *Automata Studies*, pages 3–42. Princeton University Press, Princeton, N.J., 1956.

[91] Hans-Joachim Kowalsky. Limesräume und komplettierung. *Mathematische Nachrichten*, 12:301–340, 1954.

[92] Dexter Kozen. Kleene algebra with tests. *ACM Trans. Program. Lang. Syst.*, 19(3):427–443, 1997. ISSN 0164-0925. doi: 10.1145/256167.256195.

[93] Saul A. Kripke. Semantical analysis of modal logic i normal modal propositional calculi. *Mathematical Logic Quarterly*, 9 (5-6):67–96, 1963.

[94] Werner Kuich and Arto Salomaa. *Semirings, Automata, Languages*, volume 5 of *Monographs in Theoretical Computer Science. An EATCS Series*. Springer, 1986. ISBN 3-540-13716-5.

[95] Elaine Landry and Jean-Pierre Marquis. Categories in context: Historical, foundational, and philosophical. *Philosophia Mathematica*, 13(1):1–43, 2005. doi: 10.1093/philmat/nki005.

[96] F. William Lawvere. The category of categories as a foundation for mathematics. In Eilenberg et al. [39], pages 1–20. ISBN 978-3-642-99904-8. doi: 10.1007/978-3-642-99902-4_1.

[97] Clarence I. Lewis. *A Survey of Symbolic Logic*. University of California Press, Berkeley, CA, 1918.

[98] Jacques Loeckx, Hans-Dieter Ehrich, and Markus Wolf. *Specification of Abstract Data Types*. Wiley-Teubner, 1996. ISBN 0-471-95067-X.

[99] Saunders MacLane. Natural associativity and commutativity. *Rice University Studies*, 49:28–46, 1963.

[100] Saunders MacLane. Categorical algebra. *Bull. Amer. Math. Soc. Volume*, 71(1):40–106, 1965.

[101] Saunders MacLane. *Categories for the Working Mathematician (Graduate Texts in Mathematics)*. Springer, September 1998. ISBN 0-387-98403-8.

[102] Ernest G. Manes. *Algebraic Theories*, volume 26 of *Graduate Texts in Mathematics*. Springer-Verlag, 1976.

[103] Giulio Manzonetto and Antonino Salibra. Applying universal algebra to lambda calculus. *Journal of Logic and Computation*, 20(4):877–915, 2010. doi: 10.1093/logcom/exn085.

[104] Narciso Martí-Oliet and José Meseguer. Rewriting logic as a logical and semantic framework. *Electronic Notes in Theoretical Computer Science*, 4, 1996.

[105] José Meseguer. General logics. In H.-D. Ebbinghaus, editor, *Logic Colloquium '87*, pages 275–329, Granada, Spain, July 1989. North-Holland.

[106] John Stuart Mill. *Principles of Political Economy with some of their Applications to Social Philosophy*. Longmans, Green and Co., London, 7th edition, 1909.

[107] Marvin Minsky. A framework for representing knowledge. Technical report, Cambridge, MA, USA, 1974.

[108] Eugenio Moggi. An abstract view of programming languages. Technical Report ECS-LFCS-90-113, Edinburgh Univ., Dept. of Comp. Sci., June 1989. Lecture Notes for course CS 359, Stanford Univ.

[109] Eugenio Moggi. Notions of computation and monads. *Information and Computation*, 93(1):55–92, 1991.

[110] Ben-Ari Mordechai. *Mathematical Logic for Computer Science*. Springer Verlag, 2nd edition, 2001. ISBN 1-85233-319-7.

[111] John Neumann and Oskar Morgenstern. *Theory of games and economic behavior*. Princeton University Press, Princeton, NJ, 3rd edition, 1953.

[112] Vilém Novák. On fuzzy type theory. *Fuzzy Sets and Systems*, 149(2):235–273, 2005. ISSN 0165-0114. doi: 10.1016/j.fss.2004.03.027.

[113] *Business Process Modeling Notation (BPMN) Version 1.2*. Object Management Group, January 2009.

[114] *Business Process Modeling Notation (BPMN) Version 2.0*. Object Management Group, January 2011.

[115] Frank Pfenning. Logical frameworks. In John Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*, pages 1063–1147. Elsevier and MIT Press, 2001. ISBN 0-444-50813-9, 0-262-18223-8.

[116] Jean Piaget. *The psychology of intelligence*. International library of psychology, philosophy, and scientific method. Routledge & Paul, 1950.

[117] Vaughan R. Pratt. Semantical consideration on Floyd–Hoare logic. In *Proc. 17th Annual IEEE Symposium on Foundations of Computer Science*, pages 109–121. IEEE, 1976.

[118] M. Ross Quillian. *Semantic Memory*. PhD thesis, Carnegie Institute of Technology, Pittsburgh, Pennsylvania, February 1966.

[119] M. Ross Quillian. Word concepts: a theory and simulation of some basic semantic capabilities. *Behavioral Science*, 12 (5):410–430, September 1967. ISSN 0005-7940.

[120] Gary D. Richardson. A Stone–Čech compactification for limit spaces. *Proceedings of the American Mathematical Society*, 25:403–404, 1970.

[121] David E. Rydeheard. Adjunction. In David H. Pitt, Samson Abramsky, Axel Poigné, and David E. Rydeheard, editors, *CTCS*, volume 240 of *Lecture Notes in Computer Science*, pages 51–57. Springer, 1985. ISBN 3-540-17162-2.

[122] Arto Salomaa. Two complete axiom systems for the algebra of regular events. *J. ACM*, 13(1):158–169, 1966.

[123] Manfred Schmidt-Schauß and Gert Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.

[124] Moses Schönfinkel. Über die bausteine der mathematischen logik. *Mathematische Annalen*, 92(3):305–316, 1924.

[125] Horst Schubert. *Categories*. Springer-Verlag, 1972.

[126] Dana S. Scott. Data types as lattices. *SIAM J. Comput.*, 5(3): 522–587, 1976.

[127] *Cutaneous Melanoma – A national clinical guideline*. Scottish Intercollegiate Guidelines Network, Edinburgh, July 2003.

[128] Amartya Sen. The possibility of social choice. *American Economic Review*, 89:349–378, 1999. doi: 10.1257/aer.89.3.349.

[129] M. J. Sladden, C. Balch, D. A. Barzilai, D. Berg, A. Freiman, T. Handiside, S. Hollis, M. B. Lens, and J. F. Thompson. Surgical excision margins for primary cutaneous melanoma. *Cochrane Database of Systematic Reviews*, 4, 2009. doi: 10.1002/14651858.CD004835.pub2.

[130] Lynn Arthur Steen and J. Arthur Seebach Jr. *Counterexamples in Topology*. Courier Dover Publications, September 1995. ISBN 048668735X.

[131] Lawrence Neff Stout. When does a category built on a lattice with a monoidal structure have a monoidal structure? *Fuzzy Sets and Systems*, 161(9):1162–1174, 2010. doi: 10.1016/j.fss.2009.12.018.

[132] Umberto Straccia. A fuzzy description logic. In Jack Mostow and Chuck Rich, editors, *AAAI/IAAI*, pages 594–599. AAAI Press / The MIT Press, 1998. ISBN 0-262-51098-7.

[133] Alfred Tarski. On the calculus of relations. *The Journal of Symbolic Logic*, 6(3):73–89, 1941.

[134] Endel Tulving. Episodic and semantic memory. In Endel Tulving and W. Donaldson, editors, *Organization of Memory*, pages 381–403. Academic Press, New York, 1972.

[135] Philip Wadler. Comprehending monads. *Mathematical Structures in Computer Science*, 2(4):461–493, 1992. doi: 10.1017/S0960129500001560.

[136] World Health Organization. *International Classification of Functioning, Disability and Health: ICF.* Nonserial Publication. World Health Organ, 2001. ISBN 9789241545426.

[137] John Yen. Generalizing term subsumption languages to fuzzy logic. In *IJCAI*, pages 472–477, 1991.

[138] Lotfi A. Zadeh. Fuzzy sets. *Information and Control*, 8(3): 338–353, 1965. doi: 10.1016/S0019-9958(65)90241-X.

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography *"The Elements of Typographic Style"*. `classicthesis` is available for both LaTeX and LyX:

http://code.google.com/p/classicthesis/

Happy users of `classicthesis` usually send a real postcard to the author, a collection of postcards received so far is featured here:

http://postcards.miede.de/