LOGICS AND REASONING FOR COMPUTATIONAL CREATIVITY

by

CLAUDIA ELENA CHIRIŢĂ

A thesis submitted for the degree of

Doctor of Philosophy

Department of Computer Science
Royal Holloway University of London

2018

DECLARATION

I, Claudia Elena Chiriţă, hereby declare that this thesis and the work presented in it is entirely my own. Where I have consulted the work of others, this is explicitly stated.

Claudia Elena Chiriţă
September 2018

ABSTRACT

Computational creativity is a subdomain of artificial intelligence that explores the potential of computational systems to create original artefacts and ideas. Overlapping with the broader field of cognitive science, it encompasses "the philosophy, science and engineering of computational systems which . . . exhibit behaviours that unbiased observers would deem to be creative" [CW12].

We study creativity from an algebraic point of view, showing how we can give a mathematical formalization of creative systems and their components. We start from the tenet that creativity can be seen in essence as the identification or location of new conceptual objects in a conceptual space. While most of the current research in computational creativity embraces a connectionist view on cognitive and, in particular, creative processes, our approach adheres to the symbolic computational theory of mind. We adopt the understanding of a concept as an algebraic specification, and develop our study based on Goguen and Burstall's theory of institutions. This allows us to use formal definitions for concept discovery, abstraction, concretization, and blending that enable reasoning about creative processes.

We first define creative systems by means of specifications over many-valued logics and of abstract strategies for the discovery and evaluation of concepts using the notion of graded consequence. We then focus on a subclass of creative systems modelled as complex dynamic systems and investigate a new connection between improvisation processes and service-oriented architectures where concepts and concept discovery are regarded as modules and service discovery, respectively. In this context, we evaluate the usefulness of a concept through the mechanism of service selection, and recast concept blending in terms of service binding. This leads us to the implementation of a specification and programming language whose operational semantics extends the logic-programming semantics of services from the classical Boolean setting to one with multiple truth values, akin to fuzzy logic programming.

As a case study, we model free jazz improvisations using notions and techniques from service-oriented computing. To that end, we study a suite of logics for describing music using soft constraints or specialized notations. We show how musical fragments can be captured by means of specifications of service modules, and we instantiate the service-oriented processes of discovery, selection and binding to simulate music improvisation.

ACKNOWLEDGEMENTS

I would like to thank my supervisor, José Luiz Fiadeiro, for his guidance and encouragement throughout my studies, and for giving me complete freedom in choosing the topics and methods of research. I value highly his generosity and open-mindedness.

I owe all the pleasures that have come with this academic experience to the best teacher one could ever hope to have, Ionuţ.

I am grateful for the support of my family and friends. I thank Georgiana Lungu, Răzvan and Tashi-Tsering Diaconescu, and Thomas van Binsbergen, who have been a constant source of great inspiration throughout these years and have taught me invaluable lessons on dedication, curiosity, and ethics.

I would like to thank musicians Mihai Iordache, Mircea Tiberian, Nuno Barreiro, and Shabaka Hutchings for the insightful discussions on music and improvisation.

CONTENTS

1 INTRODUCTION

1.1 COMPUTATIONAL CREATIVITY

Computational creativity is generally understood as the study of those computational activities that produce original artefacts. Although an exact formulation of what computational creativity precisely is has yet to be adopted by the members of the community involved in this area of research, there have been numerous efforts in the past decades for pinning a suitable definition; [Bod91; Sch96; Wig06a; CVW09; CW12] are only a few examples. In this thesis, we start from a prominent current understanding of computational creativity: as a subdomain of artificial intelligence that explores the potential of computational systems to create original artefacts and ideas. Overlapping with the field of cognitive science, computational creativity encompasses "the philosophy, art, science and engineering of computational systems which, by taking on particular responsibilities, exhibit behaviours that unbiased observers would deem to be creative" [CW12]. It subsumes in this way multidisciplinary research, being at the intersection of fields such as philosophy, artificial intelligence, and cognitive psychology, with an almost omnipresent desire of artistic achievement.

The goals of computational creativity – as currently stated in the field's manifesto [Icc] – are the computational modelling and simulation of creativity, as a mean for reaching one of the following objectives: the construction of software or hardware products capable of achieving human-level creativity, a better understanding of human creativity and the formulation of an algorithmic interpretation of the human creative behaviour, and the design of tools that enable and enhance human creativity.

TYPES OF COMPUTATIONAL CREATIVITY. Since the early beginnings of computational creativity as part of artificial intelligence, a distinction has been made between three types of creativity [Bod91]:

1. *exploratory creativity*, defined as the exploration of a class of possible options

(or *concepts*) within a restricted domain of interest (what is called a *conceptual space*), in order to identify or locate original conceptual objects;

2. *transformational creativity*, defined as the transformation of the class of possible options by changing the boundaries that limit the scope of the domain space, in order to accommodate new conceptual objects;
3. *combinational creativity*, defined as the combination of objects within a conceptual space for obtaining new concepts (in two words, *concept blending*).

We consider unfounded the claims that this taxonomy of computational creativity is relevant only from a purely philosophical stance: it seems that, in practice, the efforts for building creative systems can be separated into these three categories as well.

In the present thesis, we focus on the first two types of creativity, while briefly sketching a possible approach to the third type.

1.2 CREATIVE SYSTEMS

Starting from the ideas expressed in [Bod91], Geraint Wiggins proposed in [Wig06a] a mathematical formalization of the notion of *creative system*, aiming at a framework for the description, analysis and comparison of such systems. Although far from achieving this ultimate goal, the study clarifies and extends some of the ideas from computational creativity based on the state-space-search model from artificial intelligence [Wig06b]. In other words, the author abstracts over the interpretation of conceptual spaces as multidimensional spaces, over the interpretation of concepts as elements (vectors) in these spaces, and over the interpretation of the exploration of conceptual spaces as state-space search. He considers that, instead of multidimensional spaces, conceptual spaces are sets containing concepts as elements, and they can be traversed according to a given agenda. In the following chapters, we retrace some of the main ideas advanced in that paper, and define a creative system as a tuple consisting of:

- a *universe* – the space of all possible concepts,
- a *language* over which we can express rules for defining conceptual spaces as subspaces of the universe,
- a set of *rules defining conceptual spaces*,
- a function that gives *interpretations* to these rules,
- a *strategy* for the traversal of the conceptual space that, provided with a concept as input, returns another concept to be examined next, and

- a *method for evaluating* concepts in terms of usefulness, novelty, etc.

The *exploratory* kind of creativity is accommodated by this framework in a straightforward manner, through the strategy used for traversing the conceptual space. *Transformational* creativity can be modelled as the change of the rules that define a conceptual space, and thus as the morphing of the space itself in tune with the change of its interpretation, as the change of the traversal strategy, or even just as the change of the evaluation method. Lastly, *combinational* creativity is captured more subtly, through the change of the rules defining the conceptual space so as to incorporate new concepts. However, as long as concepts are only considered at the object level, it is impossible to capture in detail the process of manufacturing new concepts to be added in the conceptual space (or to be discovered in a part of the universe which does not belong to the conceptual subspace). One needs to look inside a concept, at the structural level, to be able to accurately model its morphing, as concept blending implies the recombination of the properties of two existing artefacts for the creation of a new one.

In the context of concept blending, the concepts are now generally defined as algebraic specifications[1]. However, since in this study we focus on the two other types of creativity, we adopt a more general understanding of a concept, as a specification over an arbitrary logic. This immediately raises the question of the nature of our view on cognition.

Most existing creative systems, with the exception of those belonging to the class of combinational creative systems, are based either on the connectionist view on cognitive process – implementing different types of systems derived from and mixing machine-learning techniques and neural networks, or on evolutionary algorithms. The main goal of our thesis is to investigate how creative systems can be defined in the context of symbolic models of cognition. The success of symbolic combinational creative systems based on categorical concept blending is already widely recognised – see for example, the studies that resulted from the CoInvent project [Sch+14]. We see this as an argument for modelling creative systems of exploratory and transformational nature based on similar formal, logical foundations. We choose to base our study on the theory of institutions [GB92], which was introduced by Joseph Goguen and Rod Burstall in the context of algebraic specifications.

Intuitively, we consider a creative space to be a framework based on an arbitrary logic (institution), which allows the change of the language to be

[1] We discuss concept blending in detail in Section 6.2. For now, we only give a simple, intuitive characterization.

used – defined through the symbols of a structure called *signature*. The universe comprises all possible specifications written over the logic, while a conceptual space can be seen as a subclass of these, defined by means of extensions of a given 'ground' specification. The interpretations of these spaces are given by the classes of models that satisfy the specifications. Concepts are basic specifications (presentations), and their evaluation is given by means of a notion of graded semantic entailment. The matter of exploring conceptual spaces is much more complex; to address it, we choose to study a class of such creative systems where the space exploration is achieved by means of many-valued logic programming.

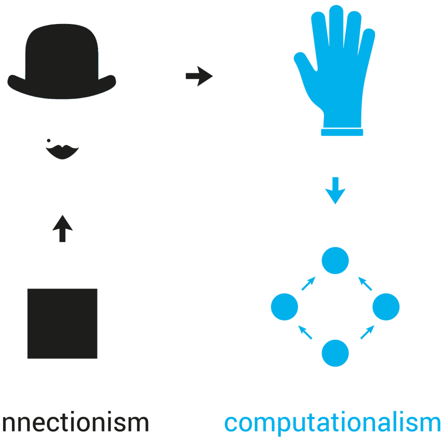


FIGURE 1.1. Bridging connectionism and computationalism

Our goal of studying creative systems under the symbolic view on cognition should be placed however in the current context of artificial intelligence, where the increased effort of reconciling connectionism and computationalism is already showing promising results [Bes+17a]. Approaches based on neural networks are evolving from black boxes which cannot provide explanations for their solutions towards bridging the gap between the two paradigms of the theory of mind: *relational networks* [San+17] and the high-level abstract representations obtained via *modular deep networks* [HOT06] are just two of many successful early results. It seems that connectionism is slowly gaining the cognitive capacity of *the man who mistook his wife for a hat*: in Oliver Sacks' famous book "The Man Who Mistook His Wife for a Hat and Other Clinical Tales" [Sac85], the curious case of a man who was suffering from visual agnosia is presented. The man could not recognise objects or persons, but his eyesight was untouched. He thus described the objects around him, structurally, in detail, trying to make sense of the world. For example, he could not recognize a glove, but he could describe it as "a continuous surface, infolded on itself, with five outpouchings". The step from accurate identification of features to realization of concepts (depicted as the horizontal arrow from Figure 1.1) is however a difficult one, especially when presented only with a small number of examples. Although solutions such as deep neural networks can somewhat lead to abstract representations of concepts, they require a very large number of input examples in the training stage. Once the task of inferring complex specifications from a set of salient features will be achieved, one could consider automatic learning from examples for defining concepts. In this thesis, we focus on the blue link in Figure 1.1: we assume that specifications of some basic concepts are already given (perhaps as a result of a learning process), and we deal with the exploration of the space to which those concepts belong, in order to derive/create new, more complex, composite concepts.

1.3 THE STRUCTURE OF THE THESIS

This thesis comprises four main chapters, each of them being devoted to technical aspects pertaining to three different levels of our approach to creativity:

In Chapter 2, we recall from the literature a series of important notions and results on category theory, institution theory and residuated lattices that we use in this study. In particular, a notable result presented in Section 2.2 is the characterization of institutions as functors into a category of Boolean rooms; this facilitates the extension of the concept of institution to a many-valued setting. A second important role of Chapter 2 is to set the basic terminology and notation.

Chapter 3 focuses on the logical foundations of the framework that we propose. These foundations are presented in three stages: first, we discuss many-valued institutions, which extend ordinary institutions in the sense that the satisfaction is no longer Boolean, but valued over a residuated lattice. Here it is worth mentioning that this belongs to a long series of developments into formalizing many-valued logics. What distinguishes our contribution from previous efforts is that we do not commit to a fixed residuated lattice, but allow instead the lattice to change along signature morphisms. The second part, Section 3.2, can be as seen as an intermediate step between service-oriented computing, algebraic specification, and creativity. Its role is to show how the use of soft constraints (as they occur in the negotiation of service level agreements in service-oriented computing) can be axiomatized by means of novel many-valued soft-constraint institutions. This development builds on previous approaches to music composition based on Boolean constraint satisfaction problems, and eases the transition towards a new logic that we specifically introduce for describing free-jazz compositions. That logic is introduced in Section 3.3, where we give a logical interpretation of the symbols used in Anthony Braxton's graphical notation for music composition.

In Chapter 4, we focus on the way in which fundamental processes from service-oriented computing can be used to generate new concepts, showing in this way how service-oriented computing has an intrinsic creative characteristic. The service-oriented processes that we consider are those through which applications discover at run time providers for the services they need, select the most suitable provider, and bind to it. To illustrate how this observation relates to music composition, in Section 4.2, we give a comprehensive example on the composition of music fragments from Steve Reich's "Clapping Music". In the conclusion of this chapter, we briefly discuss how the execution of such systems varies according to different bindings of providers, and changes in the applications' value systems.

By the nature of the construction, the framework presented in Chapter 4 is limited to service-oriented systems, where the interfaces between applications and service providers have a specific form that dictates the way in which service level agreements are determined. However, the creative potential of the systems is independent of the interfaces. To eliminate this commitment on specific interfaces, in Chapter 5, we explore a recent connection between service-oriented computing and logic programming, grounded in Boolean reasoning, and upgrade it to a many-valued setting. This raises new technical challenges, since we need to fundamentally rethink notions such as query, clause, unification and resolution. The benefit of this effort is that it provides a way to explore conceptual spaces without relying on service-oriented interfaces; moreover, it provides for free new features (morphisms of logic programs) that open the possibility to address other kinds of creativity, such as transformational creativity, which require a macroscopic view on systems.

1.4 RELEVANT PUBLICATIONS

The main ideas of the thesis and parts of it have been presented at conferences and workshops, and have been published in the following written materials:

- Sections 3.1, 3.2.1 of Chapter 3 and Sections 4.1 and 4.3 of Chapter 4 are based on the paper [CFO16], which has been presented at the 19th International Conference, FASE 2016, as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2016.
- Sections 3.2.2 and 3.3 of Chapter 3 and Section 4.2 of Chapter 4 are based on the paper [CF16], which has been presented at the Seventh International Conference on Computational Creativity, and on early results presented at the 23rd International Workshop on Algebraic Development Techniques, WADT 2016.
- A journal paper based on Chapters 3 and 4 has been submitted for publication.

2

TECHNICAL PRELIMINARIES

We reserve this chapter for revisiting some of the most relevant concepts and notations from category and institution theory to our thesis, and for presenting the algebraic structure of residuated lattices. In the first two sections, we review the concepts of comma category, factorization system, Grothendieck construction, institution and comorphism of institutions, basic specification, and model amalgamation; we also recall the presentation of institutions as functors into the category of rooms. Propositional and many-sorted first-order logic are described in detail as main examples of institutions. We then take a first step towards multivaluedness by presenting the variety of residuated lattices in a categorical context and by listing some basic properties and examples.

2.1 CATEGORIES

Throughout the thesis, we assume the reader is familiar with basic notions of category theory such as category, functor, natural transformation, and universal constructions, and we use mainly the terminology and notations from [Lan98].

For any category $\mathbb{C}$, we denote the collection of its objects by $|\mathbb{C}|$, the collection of arrows from object A to B by $\mathbb{C}(A, B)$, the composition of arrows f and g (in diagrammatic order) by $f\,;g$, and the identity arrow of an object A by id_A. In addition, for any natural transformations τ and σ, we denote by $\tau\,;\sigma$ their vertical composition, and by $\tau \cdot \sigma$ their horizontal composition. For any two categories $\mathbb{C}$ and $\mathbb{D}$, we denote by $[\mathbb{C} \to \mathbb{D}]$ the category of functors from $\mathbb{C}$ to $\mathbb{D}$, where the arrows are natural transformations and their composition is vertical. Finally, we denote by $\mathbb{S}\mathrm{et}$ the category of sets and functions, and by $\mathbb{C}\mathrm{at}$ the (quasi-)category of categories and functors.

In what follows, we recall some definitions and properties of category-theoretic concepts fundamental to this thesis. For more detailed presentations, we refer the reader both to canonical works such as [Lan98; AHS09], and to more specialised texts on the applications of category theory to algebraic specification [ST12] or software engineering [Fia05].

DEFINITION 2.1.1 (Comma category). Let $F_1\colon \mathbb{C}_1 \to \mathbb{K}$ and $F_2\colon \mathbb{C}_2 \to \mathbb{K}$ be two functors with the same codomain. The *comma category* $F_1 \,/\, F_2$ is defined as follows:

- the *objects* are triples $\langle A_1, f, A_2 \rangle$, where A_1 is an object of $\mathbb{C}_1$, A_2 is an object of $\mathbb{C}_2$, and $f\colon F_1(A_1) \to F_2(A_2)$ is an arrow in $\mathbb{K}$;
- the *arrows* $\langle A_1, f, A_2 \rangle \to \langle A'_1, f', A'_2 \rangle$ are pairs $\langle g_1, g_2 \rangle$, where $g_1\colon A_1 \to A'_1$ and $g_2\colon A_2 \to A'_2$ are arrows in $\mathbb{C}_1$ and $\mathbb{C}_2$, respectively, such that $f \mathbin{;} F_2(g_2) = F_1(g_1) \mathbin{;} f'$;
- the *composition* of arrows is defined componentwise: for every pair of arrows $\langle g_1, g_2 \rangle$ and $\langle g'_1, g'_2 \rangle$, $\langle g_1, g_2 \rangle \mathbin{;} \langle g'_1, g'_2 \rangle = \langle g_1 \mathbin{;} g'_1, g_2 \mathbin{;} g'_2 \rangle$.

$$\begin{tikzcd}
F_1(A_1) \arrow[r, "F_1(g_1)"] \arrow[d, "f"'] & F_1(A'_1) \arrow[r, "F_1(g'_1)"] \arrow[d, "f'"] & F_1(A''_1) \arrow[d, "f''"] \\
F_2(A_2) \arrow[r, "F_2(g_2)"'] & F_2(A'_2) \arrow[r, "F_2(g'_2)"'] & F_2(A''_2)
\end{tikzcd}$$

The comma category $F_1 \,/\, F_2$ is usually denoted by $\mathbb{C}_1 \,/\, F_2$, $F_1 \,/\, \mathbb{C}_2$, or simply by $\mathbb{C}_1 \,/\, \mathbb{C}_2$ if F_1, F_2, or both of them correspond to inclusions of categories. When F_1 is the constant functor with value $A \in |\mathbb{K}|$ and F_2 is the identity of $\mathbb{K}$, we denote the comma category $F_1 \,/\, F_2$ by $A \,/\, \mathbb{K}$ – the *category of* $\mathbb{K}$*-objects under* A – and the forgetful functor $A \,/\, \mathbb{K} \to \mathbb{K}$ that maps every object $\langle A, f, A' \rangle$ of $A \,/\, \mathbb{K}$ to A' by $|\_|_A$.

DEFINITION 2.1.2 (Arrow category). For any category $\mathbb{K}$, the *category* $\mathbb{K}^{\to}$ *of* $\mathbb{K}$*-arrows* is given by the comma category $\mathrm{id}_{\mathbb{K}} \,/\, \mathrm{id}_{\mathbb{K}}$.

Intuitively, the objects of the arrow category $\mathbb{K}^{\to}$ are arrows $f\colon A_1 \to A_2$ in $\mathbb{K}$, and the morphisms in $\mathbb{K}^{\to}$ from $f\colon A_1 \to A_2$ to $f'\colon A'_1 \to A'_2$ correspond to commutative squares:

$$\begin{tikzcd}
A_1 \arrow[r, "g_1"] \arrow[d, "f"'] & A'_1 \arrow[d, "f'"] \\
A_2 \arrow[r, "g_2"'] & A'_2
\end{tikzcd}$$

We denote the projection functors $\mathbb{K}^{\to} \to \mathbb{K}$ that map the arrows $f\colon A_1 \to A_2$ in $\mathbb{K}$ to their *domain* A_1 and *codomain* A_2 by $\mathrm{dom}\colon \mathbb{K}^{\to} \to \mathbb{K}$ and $\mathrm{cod}\colon \mathbb{K}^{\to} \to \mathbb{K}$, respectively.

The notion of factorization system for a category can be encountered in the literature under various definitions – equivalent or similar to a great extent. The one that we consider in this thesis is based on the original formalization from [HS73].

DEFINITION 2.1.3 (Factorization system). A *factorization system* for a category $\mathbb{C}$ consists of a pair $\langle \mathbb{E}, \mathbb{M} \rangle$ of subcategories of $\mathbb{C}$ such that

- $\mathbb{E}$ and $\mathbb{M}$ have all the objects of $\mathbb{C}$;
- all arrows in $\mathbb{E}$ are epimorphisms in $\mathbb{C}$; all arrows in $\mathbb{M}$ are monomorphisms in $\mathbb{C}$;
- all isomorphisms in $\mathbb{C}$ are in both $\mathbb{E}$ and $\mathbb{M}$;
- any morphism f of $\mathbb{C}$ can be factored as $f = e \mathbin{;} m$, with $e \in \mathbb{E}$ and $m \in \mathbb{M}$; moreover, the factorization is unique up to isomorphism, which means that for any other factorization $e' \mathbin{;} m'$ of f, with $e' \in \mathbb{E}$ and $m' \in \mathbb{M}$, there exists a unique isomorphism i in $\mathbb{C}$ such that $e \mathbin{;} i = e'$ and $i \mathbin{;} m' = m$.

$$\begin{tikzcd}
 & \bullet \arrow[dl, "e"'] \arrow[dr, "e'"] \arrow[dd, "f"] & \\
\bullet \arrow[dr, "m"'] \arrow[rr, dashed, bend left=90, "i"] & & \bullet \arrow[dl, "m'"] \\
 & \bullet &
\end{tikzcd}$$

The results presented in this work are independent of the choice of factorization of f. For this reason, assuming that A is the domain of f, we generally denote the codomain of e by $f(A)$.

FACT 2.1.4. For any categories $\mathbb{C}$ and $\mathbb{C}'$ with factorization systems $\langle \mathbb{E}, \mathbb{M} \rangle$ and $\langle \mathbb{E}', \mathbb{M}' \rangle$, respectively, we can obtain the following factorization systems:

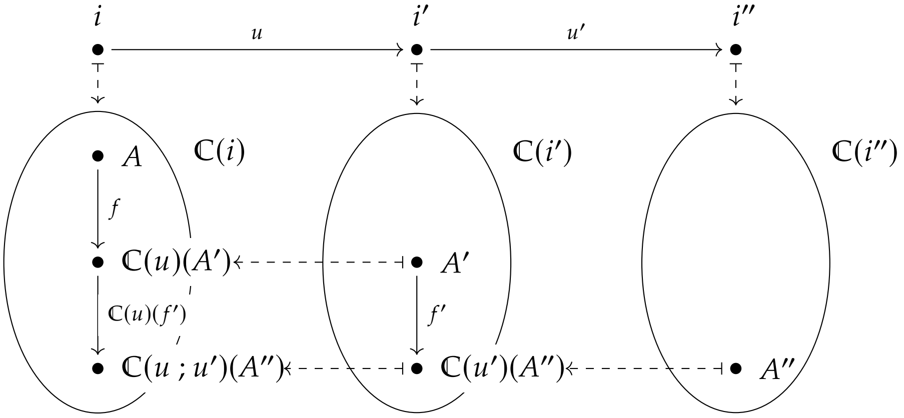
1. $\langle \mathbb{M}, \mathbb{E} \rangle$ for the dual category $\mathbb{C}^{\mathrm{op}}$,
2. $\langle C/\mathbb{E}, C/\mathbb{M} \rangle$ for the comma category $C/\mathbb{C}$, where $C \in |\mathbb{C}|$,
3. $\langle \mathbb{E} \times \mathbb{E}', \mathbb{M} \times \mathbb{M}' \rangle$ for the product category $\mathbb{C} \times \mathbb{C}'$.

DEFINITION 2.1.5 (Indexed category). Given a category $\mathbb{I}$ of *indices*, an $\mathbb{I}$-*indexed category* is a functor $\mathbb{C} \colon \mathbb{I}^{\mathrm{op}} \to \mathbb{C}\mathrm{at}$.

DEFINITION 2.1.6 (Grothendieck construction). Any $\mathbb{I}$-indexed category $\mathbb{C} \colon \mathbb{I}^{\mathrm{op}} \to \mathbb{C}\mathrm{at}$ can be flattened to a *Grothendieck category* $\mathbb{C}^{\#}$ whose

- *objects* are pairs $\langle i, A \rangle$, where i is an object of $\mathbb{I}$ and A is an object of $\mathbb{C}(i)$,
- *arrows* $\langle i, A \rangle \to \langle i', A' \rangle$ are pairs $\langle u, f \rangle$ such that $u \colon i \to i'$ is an arrow in $\mathbb{I}$ and $f \colon A \to \mathbb{C}(u)(A')$ is an arrow in $\mathbb{C}(i)$, and

- *composition of arrows* $\langle u, f\rangle$ and $\langle u', f'\rangle$ is defined as $\langle u\,;u', f\,;\mathbb{C}(u)(f')\rangle$.



Fact 2.1.7. For any indexed category $\mathbb{C}\colon \mathbb{I}^{\mathrm{op}} \to \mathbb{C}\mathrm{at}$, the Grothendieck category $\mathbb{C}^{\#}$ is the vertex of the lax colimit[1] $\mu\colon \mathbb{C} \to \mathbb{C}^{\#}$ of $\mathbb{C}$ in $\mathbb{C}\mathrm{at}$, where:

- for every index $i \in |\mathbb{I}|$, $\mu^{i}\colon \mathbb{C}(i) \to \mathbb{C}^{\#}$ is an embedding of categories, and
- for every morphism of indexes $u\colon i' \to i \in \mathbb{I}^{\mathrm{op}}$, $\mu^{u}\colon \mathbb{C}(u)\,;\mu^{i} \Rightarrow \mu^{i'}$ is defined by $\mu^{u}_{A'} = \langle u, \mathrm{id}_{\mathbb{C}(u)(A')}\rangle$ for every object $A' \in |\mathbb{C}(i')|$.

Fact 2.1.8. Every *indexed functor* F between the $\mathbb{I}$-indexed categories $\mathbb{C}$ and $\mathbb{D}\colon \mathbb{I}^{\mathrm{op}} \to \mathbb{C}\mathrm{at}$, i.e., every natural transformation $F\colon \mathbb{C} \Rightarrow \mathbb{D}$, determines a functor $F^{\#}\colon \mathbb{C}^{\#} \to \mathbb{D}^{\#}$ that maps

- every object $\langle i, A\rangle$ of the Grothendieck category $\mathbb{C}^{\#}$ to $\langle i, F_i(A)\rangle$ and
- every morphism $\langle u, f\rangle\colon \langle i, A\rangle \to \langle i', A'\rangle$ in $\mathbb{C}^{\#}$ to $\langle u, F_i(f)\rangle$.

We obtain a flattening functor $(\_)^{\#}$ from the category $[\mathbb{I}^{\mathrm{op}} \to \mathbb{C}\mathrm{at}]$ of $\mathbb{I}$-indexed categories to $\mathbb{C}\mathrm{at}$.

Definition 2.1.9 (Category of functors). For any category $\mathbb{K}$, the category $[\_ \to \mathbb{K}]^{\#}$ of *functors into* $\mathbb{K}$ is the Grothendieck category $[\_ \to \mathbb{K}]\colon \mathbb{C}\mathrm{at}^{\mathrm{op}} \to \mathbb{C}\mathrm{at}$ that maps every category $\mathbb{C}$ to the functor category $[\mathbb{C} \to \mathbb{K}]$ and any functor $U\colon \mathbb{C} \to \mathbb{C}'$ to the functor $U\_\colon [\mathbb{C}' \to \mathbb{K}] \to [\mathbb{C} \to \mathbb{K}]$:

- the *objects* of $[\_ \to \mathbb{K}]^{\#}$ are functors $F\colon \mathbb{C} \to \mathbb{K}$,
- the *morphisms* between $F\colon \mathbb{C} \to \mathbb{K}$ and $F'\colon \mathbb{C}' \to \mathbb{K}$ are pairs $\langle U, \rho\rangle$ where $U\colon \mathbb{C} \to \mathbb{C}'$ is a functor, and $\rho\colon F \Rightarrow U\,;F'$ is a natural transformation. The composition of arrows is defined by $\langle U, \rho\rangle\,;\langle U', \rho'\rangle = \langle U\,;U', \rho\,;(U \cdot \rho')\rangle$.

[1] Lax colimits [Bor94; Dia11] are a more relaxed concept of colimit in 2-categories, where the strict equality requirement is replaced with diagram commutativity up to 2-cells.

FACT 2.1.10. Functors $G\colon \mathbb{K} \to \mathbb{K}'$ determine appropriate natural transformations $[G]\colon [\_ \to \mathbb{K}] \Rightarrow [\_ \to \mathbb{K}']$ having as components $[G]_{\mathbb{C}}$ for any category $\mathbb{C} \in |\mathbb{C}\mathrm{at}|$ the right-composition functors from $[\mathbb{C} \to \mathbb{K}]$ to $[\mathbb{C} \to \mathbb{K}']$. Hence, every functor $G\colon \mathbb{K} \to \mathbb{K}'$ induces a functor $[G]^{\#}\colon [\_ \to \mathbb{K}]^{\#} \to [\_ \to \mathbb{K}']^{\#}$.

2.2 INSTITUTIONS

The notion of institution arose in the field of algebraic specification to answer the need for a framework for studying concepts and techniques related to structuring and modularising specifications independently of the languages used. It was introduced by Goguen and Burstall in [GB83; GB92] to formalize the intuitive notion of logical system as a mediated relation between its syntax and semantics. Offering a rigorous mathematical abstraction of the main ingredients of logics, institution theory proved to be suitable for developing the foundations of specification languages.

In short, an institution consists of a collection (category) of so-called signatures that determine (through functors) sets of sentences and collections of models, as well as satisfaction relations between models and sentences; the satisfaction relations are assumed to be invariant under non-logical language translation (change of signature).

DEFINITION 2.2.1 (Institution). An *institution* $\mathfrak{I}$ consists of

- a category $\mathbb{S}\mathrm{ig}^{\mathfrak{I}}$ of *signatures* and *signature morphisms*,
- a *sentence functor* $\mathrm{Sen}^{\mathfrak{I}}\colon \mathbb{S}\mathrm{ig}^{\mathfrak{I}} \to \mathbb{S}\mathrm{et}$ giving, for every signature Σ, the set $\mathrm{Sen}^{\mathfrak{I}}(\Sigma)$ of Σ-*sentences* and, for every signature morphism $\varphi\colon \Sigma \to \Sigma'$, the *sentence translation map* $\mathrm{Sen}^{\mathfrak{I}}(\varphi)\colon \mathrm{Sen}^{\mathfrak{I}}(\Sigma) \to \mathrm{Sen}^{\mathfrak{I}}(\Sigma')$,
- a *model functor* $\mathrm{Mod}^{\mathfrak{I}}\colon (\mathbb{S}\mathrm{ig}^{\mathfrak{I}})^{\mathrm{op}} \to \mathbb{C}\mathrm{at}$ defining, for every signature Σ, the category $\mathrm{Mod}^{\mathfrak{I}}(\Sigma)$ of Σ-*models* and Σ-*model homomorphisms* and, for every signature morphism $\varphi\colon \Sigma \to \Sigma'$, the *reduct functor* $\mathrm{Mod}^{\mathfrak{I}}(\varphi)$ that maps models over Σ' to models over Σ,
- a *satisfaction relation* $\vDash^{\mathfrak{I}}_{\Sigma} \subseteq |\mathrm{Mod}^{\mathfrak{I}}(\Sigma)| \times \mathrm{Sen}^{\mathfrak{I}}(\Sigma)$ for every signature Σ determining the satisfaction of Σ-sentences by Σ-models,

such that the *satisfaction condition*

$$M' \vDash^{\mathfrak{I}}_{\Sigma'} \mathrm{Sen}^{\mathfrak{I}}(\varphi)(\rho) \qquad \text{if and only if} \qquad \mathrm{Mod}^{\mathfrak{I}}(\varphi)(M') \vDash^{\mathfrak{I}}_{\Sigma} \rho$$

holds for any signature morphism $\varphi\colon \Sigma \to \Sigma'$, Σ'-model M', and Σ-sentence ρ. Intuitively, this means that the translation of sentences $\mathrm{Sen}^{\mathfrak{I}}(\varphi)$ and the reduction of models $\mathrm{Mod}^{\mathfrak{I}}(\varphi)$ preserve the satisfaction relation.

REMARK 2.2.2. The satisfaction relation corresponding to any signature Σ of an institution $\mathcal{I}$ can also be presented as a function $\models^{\mathcal{I}}_{\Sigma} \colon \mathrm{Sen}^{\mathcal{I}}(\Sigma) \to [|\mathrm{Mod}^{\mathcal{I}}(\Sigma)| \to \mathbb{2}]$ that determines if any given Σ-sentence is satisfied by any given Σ-model. In this way, we can encapsulate the satisfaction condition in a categorical construct by defining the relations $\models^{\mathcal{I}}_{\Sigma}$ as the components of a natural transformation $\models^{\mathcal{I}} \colon \mathrm{Sen}^{\mathcal{I}} \Rightarrow [|\mathrm{Mod}^{\mathcal{I}}(\_)| \to \mathbb{2}]$ between the sentence functor and the functor $[|\mathrm{Mod}^{\mathcal{I}}(\_)| \to \mathbb{2}]$ assigning to every signature Σ the collection of functions from $|\mathrm{Mod}^{\mathcal{I}}(\_)|$ to the values of the Boolean lattice $\mathbb{2}$.

$$\begin{tikzcd}
\Sigma \arrow[d, "\varphi"'] & \mathrm{Sen}^{\mathcal{I}}(\Sigma) \arrow[r, "\models^{\mathcal{I}}_{\Sigma}"] \arrow[d, "\mathrm{Sen}^{\mathcal{I}}(\varphi)"'] & {[|\mathrm{Mod}^{\mathcal{I}}(\Sigma)| \to \mathbb{2}]} \arrow[d, "\mathrm{Mod}^{\mathcal{I}}(\varphi)\,;\,\_"] \\
\Sigma' & \mathrm{Sen}^{\mathcal{I}}(\Sigma') \arrow[r, "\models^{\mathcal{I}}_{\Sigma'}"'] & {[|\mathrm{Mod}^{\mathcal{I}}(\Sigma')| \to \mathbb{2}]}
\end{tikzcd}$$

We may omit sub- or super-scripts from the above notations of the institution constituents when there is no risk of confusion. If, for example, the institution $\mathcal{I}$ and the signature Σ are easy to infer, we denote the satisfaction relation $\models^{\mathcal{I}}_{\Sigma}$ simply by $\models$. For simplicity, we often denote the sentence translation $\mathrm{Sen}^{\mathcal{I}}(\varphi)$ by $\varphi(\_)$ and the reduct functor $\mathrm{Mod}^{\mathcal{I}}(\varphi)$ by $\_\!\upharpoonright_{\varphi}$. When $M = M'\!\upharpoonright_{\varphi}$ we say that M is the *φ-reduct* of M' and that M' is a *φ-expansion* of M.

SEMANTIC CONSEQUENCE. The satisfaction relation extends naturally to sets of sentences: a Σ-model M satisfies a set of Σ-sentences E, denoted $M \models^{\mathcal{I}}_{\Sigma} E$, if M satisfies all sentences $\rho \in E$. In addition, we say that a Σ-sentence ρ is a *semantic consequence* of a set of Σ-sentences E, denoted by $E \models^{\mathcal{I}}_{\Sigma} \rho$, when every Σ-model that satisfies every sentence in E satisfies ρ as well. Finally, we write $E \models^{\mathcal{I}}_{\Sigma} E'$, where E and E' are sets of Σ-sentences, when $E \models^{\mathcal{I}}_{\Sigma} \rho$ for every $\rho \in E'$.

EXAMPLES. The literature on specification languages and abstract model theory contains a multitude of examples of logical systems formalized as institutions, both from computer science and from mathematical logic: first-order logic [GB92], order-sorted Horn-clause logic [GM87], equational logic [Dia95], temporal logic [FC96], modal logic [DS07], many-valued logics [Dia13], hybrid logic [DM16]; see the books [ST12; Dia08] for many other detailed examples.

In what follows, we present the institutions of two logical systems of reference for our work: propositional and many-sorted first-order logic.

THE INSTITUTION $\mathcal{PL}$ OF PROPOSITIONAL LOGIC

SIGNATURES. A *propositional signature* Σ is a set whose elements are called propositional symbols or variables. *Signature morphisms* $\varphi\colon \Sigma \to \Sigma'$ are functions.

SENTENCES. Given a propositional signature Σ, its *sentences* are defined by the grammar

$$\rho ::= p \mid \top \mid \bot \mid \neg\rho \mid \rho \wedge \rho \mid \rho \vee \rho \mid \rho \to \rho \mid \rho \leftrightarrow \rho,$$

where $p \in \Sigma$. For any function $\varphi\colon \Sigma \to \Sigma'$, the *translation of a sentence* ρ along φ, $\mathrm{Sen}(\varphi)(\rho)$ is the sentence obtained by replacing in ρ every propositional symbol $p \in \Sigma$ with the symbol $\varphi(p)$.

MODELS. Given a propositional signature Σ, a Σ-*model* M is a valuation (function) from Σ to the values of the Boolean lattice $\mathbb{2}$; a model M can be extended to sentences to define $M^{\#}\colon \mathrm{Sen}(\Sigma) \to \mathbb{2}$ using the usual truth-tables associated with the propositional connectives.

For any function $\varphi\colon \Sigma \to \Sigma'$, the reduct of a Σ'-model $M'\colon \Sigma' \to \mathbb{2}$ is just the composition $(\varphi\,;M')\colon \Sigma \to \mathbb{2}$, i.e. propositional symbols in Σ are assigned the truth values associated with their images under φ.

SATISFACTION. A model M *satisfies* a sentence ρ if and only if $M^{\#}(\rho) = \mathit{true}$.

Finally, the satisfaction condition holds because, for every Σ'-model M' and every morphism $\varphi\colon \Sigma \to \Sigma'$, $(\varphi\,;M')^{\#} = \varphi\,;M'^{\#}$:

$$\begin{aligned} M'{\restriction_\varphi} \models_\Sigma \rho \quad &\text{iff} \quad (M'{\restriction_\varphi})^{\#}(\rho) = \mathit{true} \\ &\text{iff} \quad (\varphi\,;M')^{\#}(\rho) = \mathit{true} \\ &\text{iff} \quad M'^{\#}(\varphi(\rho)) = \mathit{true} \\ &\text{iff} \quad M' \models_\Sigma \varphi(\rho). \end{aligned}$$

THE INSTITUTION $\mathcal{FOL}$ OF MANY-SORTED FIRST-ORDER LOGIC

SIGNATURES. A *first-order signature* is a tuple $\langle S, F, P\rangle$ comprising:

- a set S of *sorts*,
- a family $F = \{F_{w\to s} \mid w \in S^*, s \in S\}$ of sets of *operation symbols* indexed by arities and sorts, where $F_{w\to s}$ denotes the set of operation symbols with arity

w and sort s (when the arity is empty, $F_{\epsilon\to s}$ denotes the set of *constants* of sort s), and

- a family $P = \{P_w \mid w \in S^*\}$ of sets of S-sorted *relation symbols* indexed by arities. When all P_w are empty, what we obtain is an *algebraic signature*.

The *signature morphisms* $\varphi\colon \langle S, F, P\rangle \to \langle S', F', P'\rangle$ reflect the structure of signatures and consist of

- a function $\varphi^{\mathrm{st}}\colon S \to S'$ between the sets of sorts,
- a family of functions $\varphi^{\mathrm{op}} = \{\varphi^{\mathrm{op}}_{w\to s}\colon F_{w\to s} \to F'_{\varphi^{\mathrm{st}}(w)\to\varphi^{\mathrm{st}}(s)} \mid w \in S^*, s \in S\}$ between the sets of operation symbols, and
- a family of functions $\varphi^{\mathrm{rel}} = \{\varphi^{\mathrm{rel}}_w\colon P_w \to P'_{\varphi^{\mathrm{st}}(w)} \mid w \in S^*\}$ between the sets of relation symbols.

SENTENCES. The sentences are usual first-order sentences built from equational and relational atoms by applying in an iterative manner Boolean connectives and first-order quantifiers. Given a signature $\langle S, F, P\rangle$, and a sort s, the set $T_{F,s}$ of F-terms of sort s is the least set such that $\sigma(\bar{t}) \in T_{F,s}$, for all $\sigma \in F_{w\to s}$, and all tuples $\bar{t} \in T_{F,w}$, where $T_{F,w} = T_{F,s_1} \times \cdots \times T_{F,s_n}$ for $w = s_1 \cdots s_n$. The set $\mathrm{Sen}(S, F, P)$ of $\langle S, F, P\rangle$-sentences is the least set containing *equational atoms* $t =_s t'$ (for $t, t' \in T_{F,s}$) and *relational atoms* $\pi(\bar{t})$ (where $\pi \in P_w$ and $\bar{t} \in T_{F,w}$) that is closed under

- *Boolean connectives*: for any $\langle S, F, P\rangle$-sentences ρ_1 and ρ_2, the *negation* $\neg\rho_i$, the *conjunction* $\rho_1 \wedge \rho_2$, the *disjunction* $\rho_1 \vee \rho_2$, the *implication* $\rho_1 \to \rho_2$, and the *equivalence* $\rho_1 \leftrightarrow \rho_2$ are also $\langle S, F, P\rangle$-sentences,
- *existential and universal quantification* over sets of first-order variables, which are triples $(x, s, \langle S, F, P\rangle)$ usually denoted by $x : s$ or simply by x, where x is the name of the variable, and $s \in S$ is its sort: For any finite set X of $\langle S, F, P\rangle$-variables, let $\langle S, F \uplus X, P\rangle$ be the extension of $\langle S, F, P\rangle$ with the elements of X added as new constants. Then for any $\langle S, F \uplus X, P\rangle$-sentence ρ, $\exists X.\rho$ and $\forall X.\rho$ are $\langle S, F, P\rangle$-sentences.

The *translation of sentences* $\mathrm{Sen}(\varphi)\colon \mathrm{Sen}(S, F, P) \to \mathrm{Sen}(S', F', P')$ along a signature morphism $\varphi\colon \langle S, F, P\rangle \to \langle S', F', P'\rangle$ is defined inductively on the structure of the sentences, and renames the sorts, function, and relation symbols of $\langle S, F, P\rangle$ with symbols of $\langle S', F', P'\rangle$ according to φ. For terms, we define the extension of φ as $\varphi^{\mathrm{tm}}(\sigma(\bar{t})) = \varphi^{\mathrm{op}}(\sigma)(\varphi^{\mathrm{tm}}(\bar{t}))$. Then,

- $\mathrm{Sen}(\varphi)(t =_s t') = (\varphi^{\mathrm{tm}}(t) =_{\varphi^{\mathrm{st}}(s)} \varphi^{\mathrm{tm}}(t'))$ for equations,
- $\mathrm{Sen}(\varphi)(\pi(\bar{t})) = \varphi^{\mathrm{rel}}(\pi)(\varphi^{\mathrm{tm}}(\bar{t}))$ for relational atoms,

- $\mathrm{Sen}(\varphi)(\rho_1 \wedge \rho_2) = \mathrm{Sen}(\varphi)(\rho_1) \wedge \mathrm{Sen}(\varphi)(\rho_2)$, and similarly for the rest of the Boolean connectives, and
- $\mathrm{Sen}(\varphi)(\exists X.\rho) = \exists X^{\varphi}.\mathrm{Sen}(\varphi^X)(\rho)$ for every finite set of variables X and every $\langle S, F \uplus X, P\rangle$-sentence ρ, where $X^{\varphi} = \{x : \varphi^{\mathrm{st}}(s) \mid x : s \in X\}$ and $\varphi^X \colon \langle S, F \uplus X, P\rangle \to \langle S', F' \uplus X^{\varphi}, P'\rangle$ extends φ canonically. We define the translation of universally quantified sentences in a similar manner.

MODELS. For every signature $\langle S, F, P\rangle$, a model M interprets each sort s as a set M_s, called the *carrier set* of that sort, each operation symbol $\sigma \in F_{w\to s}$ as a function $M_\sigma \colon M_w \to M_s$, where $M_w = M_{s_1} \times \cdots \times M_{s_n}$ for $w = s_1 \cdots s_n$, and each relation symbol $\pi \in P_w$ as a subset $M_\pi \subseteq M_w$.

A *homomorphism* of $\langle S, F, P\rangle$-models $h \colon M \to N$ is an indexed family of functions $\{h_s \colon M_s \to N_s \mid s \in S\}$ such that

- h is an $\langle S, F\rangle$-algebra homomorphism: $h_s(M_\sigma(m)) = N_\sigma(h_w(m))$, for every $\sigma \in F_{w\to s}$ and every $m \in M_w$, where $h_w \colon M_w \to N_w$ denotes the canonical component-wise extension of h to w-tuples, i.e. $h_w(m_1, \ldots, m_n) = (h_{s_1}(m_1), \ldots, h_{s_n}(m_n))$ for $w = s_1 \cdots s_n$ and $m_i \in M_{s_i}$ for $i = \overline{1, n}$, and
- $h_w(m) \in N_\pi$ if $m \in M_\pi$, i.e. $h_w(M_\pi) \subseteq N_\pi$, for every relation symbol $\pi \in P_w$.

For every signature morphism $\varphi \colon \langle S, F, P\rangle \to \langle S', F', P'\rangle$, the *model reduct* $M'\!\upharpoonright_{\varphi}$ of a $\langle S', F', P'\rangle$-model M' is defined as the $\langle S, F, P\rangle$-model given by $(M'\!\upharpoonright_{\varphi})_x = M'_{\varphi(x)}$ for every sort, function, or relation symbol x from the domain signature of φ. The reduct $h'\!\upharpoonright_{\varphi}$ of a model homomorphism is also defined as $(h'\!\upharpoonright_{\varphi})_s = h'_{\varphi(s)}$, for every sort $s \in S$.

SATISFACTION. The *satisfaction* between models and sentences is the usual Tarskian satisfaction defined inductively on the structure of sentences and based on the valuation of terms in models. This valuation can be defined by induction on the structure of the terms; for any term $t : s$, we denote its interpretation in a model M (a value in M_s) as M_t. Given a model M

- for equational atoms: $M \vDash_{\langle S,F,P\rangle} t =_s t'$ if $M_t = M_{t'}$,
- for relational atoms: $M \vDash_{\langle S,F,P\rangle} \pi(\overline{t})$ if $M_{\overline{t}} \in M_\pi$,[2]
- $M \vDash_{\langle S,F,P\rangle} \neg\rho$ if $M \nvDash_{\langle S,F,P\rangle} \rho$,
- $M \vDash_{\langle S,F,P\rangle} \rho_1 \wedge \rho_2$ if $M \vDash_{\langle S,F,P\rangle} \rho_1$ and $M \vDash_{\langle S,F,P\rangle} \rho_2$,
- $M \vDash_{\langle S,F,P\rangle} \rho_1 \vee \rho_2$ if $M \vDash_{\langle S,F,P\rangle} \rho_1$ or $M \vDash_{\langle S,F,P\rangle} \rho_2$,
- $M \vDash_{\langle S,F,P\rangle} \rho_1 \to \rho_2$ if $M \vDash_{\langle S,F,P\rangle} \rho_2$ whenever $M \vDash_{\langle S,F,P\rangle} \rho_1$,

[2] Note that π can be nullary; its interpretation is either the empty set, or a set with only one element, the empty tuple – in that case $\pi()$, which can be simplified to π, is satisfied by M.

- $M \models_{\langle S,F,P\rangle} \exists X.\rho$ if there exists an expansion M' of M along the signature inclusion $\langle S,F,P\rangle \hookrightarrow \langle S,F \uplus X,P\rangle$ such that $M' \models_{\langle S,F\uplus X,P\rangle} \rho$, and
- $M \models_{\langle S,F,P\rangle} \forall X.\rho$ if $M' \models_{\langle S,F\uplus X,P\rangle} \rho$ for all expansions M' of M along the signature inclusion $\langle S,F,P\rangle \hookrightarrow \langle S,F \uplus X,P\rangle$.

SPECIFICATIONS OVER INSTITUTIONS

DEFINITION 2.2.3 (Basic specification over an institution). A *basic specification*, or *(theory) presentation*, in an institution $\mathcal{I}$ is a pair $\langle \Sigma, E\rangle$ consisting of a signature Σ and a collection of sentences (axioms) $E \subseteq \mathrm{Sen}^{\mathcal{I}}(\Sigma)$ in the language of that signature. The denotational semantics of $\langle \Sigma, E\rangle$ is given by the class $\mathrm{Mod}^{\mathcal{I}\text{-pres}}(\Sigma, E)$ of Σ-models that satisfy all the sentences in E. In symbols:

$$\mathrm{Mod}^{\mathcal{I}\text{-pres}}(\Sigma, E) = \{M \in |\mathrm{Mod}^{\mathcal{I}}(\Sigma)| \mid M \models^{\mathcal{I}}_{\Sigma} E\}.$$

A *morphism* between basic specifications $\langle \Sigma, E\rangle$ and $\langle \Sigma', E'\rangle$ is a signature morphism $\varphi\colon \Sigma \to \Sigma'$ such that $E' \models^{\mathcal{I}}_{\Sigma'} \varphi(E)$, i.e. the translations along φ of the axioms of $\langle \Sigma, E\rangle$ to the language of Σ' are semantic consequences of $\langle \Sigma', E'\rangle$. Such a morphism formalizes the way in which $\langle \Sigma, E\rangle$ is a part of $\langle \Sigma', E'\rangle$. Presentations and their morphisms constitute a category, which we denote by $\mathbb{P}\mathrm{res}^{\mathcal{I}}$.

MODEL AMALGAMATION

Model amalgamation is one of the most important properties of an institution, with numerous applications in the context of institution-independent model theory [Dia08] and module algebra [DGS93]. Essentially, it allows us to combine models of different signatures whenever they are compatible with respect to a common 'sub-signature'. Many logical systems of interest in specification theory have the model-amalgamation property, including the examples considered in this thesis.

DEFINITION 2.2.4 (Model amalgamation). In any institution $\mathcal{I}$, a commuting square of signature morphisms

$$\begin{tikzcd} \Sigma \arrow[r, "\varphi_1"] \arrow[d, "\varphi_2"'] & \Sigma_1 \arrow[d, "\theta_1"] \\ \Sigma_2 \arrow[r, "\theta_2"'] & \Sigma' \end{tikzcd}$$

is a *weak amalgamation square* if, for every Σ_1-model M_1 and Σ_2-model M_2 such that $\mathrm{Mod}^{\mathcal{I}}(\varphi_1)(M_1) = \mathrm{Mod}^{\mathcal{I}}(\varphi_2)(M_2)$, there exists a Σ'-model M', called an *amalgamation* of M_1 and M_2, such that $\mathrm{Mod}^{\mathcal{I}}(\theta_1)(M') = M_1$ and $\mathrm{Mod}^{\mathcal{I}}(\theta_2)(M') = M_2$. When M' is unique with this property, the above square is called an *amalgamation square*.

We say that an institution has (weak) model amalgamation if and only if every pushout square of signature morphisms is a (weak) amalgamation square.

Therefore, in many concrete examples of institutions (like $\mathcal{PL}$ and $\mathcal{FOL}$), in order to have model amalgamation, the signature morphisms θ_1 and θ_2 must not identify entities of Σ_1 and Σ_2 that do not have a common preimage in Σ under the signature morphisms φ_1 and φ_2. Moreover, to guarantee the uniqueness of the amalgamation, Σ' must contain only entities that originate from Σ_1 or Σ_2.

MOVING BETWEEN INSTITUTIONS

When describing and reasoning about properties of highly complex structures or systems, it is often desirable to use different formalisms for different tasks. Consequently, in order to use institutions as formalizations of logical systems in a heterogeneous setting, one needs to define formally a notion of map between institutions. Several concepts have been defined over the years, including semi-morphisms, morphisms, and comorphisms, some of which can be found in [ST12]. In what follows, we will make use of comorphisms [GR02], which reflect the intuition of embedding simpler institutions into more complex ones. They were first presented in [Mes89] as *plain maps*, and in [Tar95] as *institution representations*.

DEFINITION 2.2.5 (Institution comorphism). Given two institutions $\mathcal{I} = \langle \mathbb{S}\mathrm{ig}^{\mathcal{I}}, \mathrm{Sen}^{\mathcal{I}}, \mathrm{Mod}^{\mathcal{I}}, \models^{\mathcal{I}} \rangle$ and $\mathcal{I}' = \langle \mathbb{S}\mathrm{ig}^{\mathcal{I}'}, \mathrm{Sen}^{\mathcal{I}'}, \mathrm{Mod}^{\mathcal{I}'}, \models^{\mathcal{I}'} \rangle$, an *institution comorphism* $\langle \Phi, \alpha, \beta \rangle \colon \mathcal{I} \to \mathcal{I}'$ consists of:

- a signature functor $\Phi\colon \mathbb{S}\mathrm{ig}^{\mathcal{I}} \to \mathbb{S}\mathrm{ig}^{\mathcal{I}'}$,
- a natural transformation $\alpha\colon \mathrm{Sen}^{\mathcal{I}} \Rightarrow \Phi \,;\mathrm{Sen}^{\mathcal{I}'}$, and
- a natural transformation $\beta\colon \Phi^{\mathrm{op}} \,;\mathrm{Mod}^{\mathcal{I}'} \Rightarrow \mathrm{Mod}^{\mathcal{I}}$

such that the following *satisfaction condition* holds for any $\mathcal{I}$-signature Σ, $\Phi(\Sigma)$-model M', and Σ-sentence ρ:

$$M' \models^{\mathcal{I}'}_{\Phi(\Sigma)} \alpha_\Sigma(\rho) \qquad \text{if and only if} \qquad \beta_\Sigma(M') \models^{\mathcal{I}}_{\Sigma} \rho.$$

DEFINITION 2.2.6 ($\mathbb{coIns}$). Institution comorphisms can be composed componentwise. The composition has identities and is associative, giving rise to the category $\mathbb{coIns}$ of institutions and institution comorphisms.

EXAMPLE 2.2.7 (Embedding $\mathcal{PL}$ in $\mathcal{FOL}$). Propositional logic can be seen as a fragment of first-order logic determined by signatures with empty sets of sort symbols (up to an equivalence of logical systems). To formally capture this embedding, we define an institution comorphism $\langle \Phi, \alpha, \beta \rangle \colon \mathcal{PL} \to \mathcal{FOL}$ as follows:

- Every propositional signature Σ is mapped to the first-order signature with the empty set of sorts, without function symbols, and containing only nullary predicate symbols,

$$\Phi(\Sigma) = \langle S = \emptyset, F = \emptyset, P = \Sigma \rangle$$

- The sentence translation is an inclusion – every propositional sentence is also a first-order sentence,

$$\alpha_\Sigma(\rho) = \rho$$

- Every model $M' \in |\mathrm{Mod}^{\mathcal{FOL}}(\Phi(\Sigma))|$ is trivially reduced to a Σ-model in $\mathcal{PL}$.

$$\beta_\Sigma(M')(p) = \begin{cases} \mathit{true}, & \text{if } M'_p = M'_\epsilon = \{()\} \\ \mathit{false}, & \text{if } M'_p = \emptyset \end{cases}$$

EXAMPLE 2.2.8 (Embedding $\mathcal{EQL}$ and $\mathcal{REL}$ in $\mathcal{FOL}$). In a similar way, we can present *equational logic* and *relational logic* as fragments of first-order logic.

1. The institution $\mathcal{EQL}$ of equational logic can be obtained from $\mathcal{FOL}$ by discarding the relation symbols and their interpretations. We define the comorphism $\langle \Phi, \alpha, \beta \rangle \colon \mathcal{EQL} \to \mathcal{FOL}$ as follows:

- $\Phi \colon \mathbb{Sig}^{\mathcal{EQL}} \to \mathbb{Sig}^{\mathcal{FOL}}$ is the embedding of algebraic structures into the category of first-order signatures,
- for every signature $\Sigma \in |\mathbb{Sig}^{\mathcal{EQL}}|$, α_Σ is an inclusion of sets, and β_Σ is the identity functor.

2. The institution $\mathcal{REL}$ of relational logic can be seen as a sub-institution of $\mathcal{FOL}$ determined by the class of signatures without non-constant operation symbols. We can define a comorphism $\langle \Phi, \alpha, \beta \rangle \colon \mathcal{REL} \to \mathcal{FOL}$ similarly to the previous example.

EXAMPLE 2.2.9 (Encoding relations as operations). A less trivial example of an institution comorphism is the encoding of $\mathcal{REL}$ into $\mathcal{EQL}$ by representing relations as Boolean-valued operations. The comorphism $\langle \Phi, \alpha, \beta \rangle \colon \mathcal{REL} \to \mathcal{EQL}$

- maps every relational signature $\langle S, F, P \rangle$ to the algebraic signature $\langle S \uplus \{bool\}, F \uplus \{true\} \uplus P^{op} \rangle$, where *bool* is a new sort, *true* is a new constant of sort *bool*, and for every $w \in S^*$, $P^{op}_{w \to s} = P_w$, for $s = bool$, and $P^{op}_{w \to s} = \emptyset$, otherwise,
- maps every relational atom $\pi(t)$ to the equation $\pi(t) = true$, and
- maps every $\langle S \uplus \{bool\}, F \uplus \{true\} \uplus P^{op} \rangle$-algebra M' to the $\langle S, F, P \rangle$-model $\beta(M')$ that has the same carrier sets for the sorts in S and the same interpretations of the operation symbols in F, and for every relation symbol π, $\beta(M')_\pi = (M'_\pi)^{-1}(M'_{true})$.

INSTITUTIONS AS FUNCTORS

Following the idea in Remark 2.2.2, we can alternatively present an institution by giving for every signature a set of sentences, a category of models, and a function mapping every sentence to a valuation function that assigns to every model a truth value from $\mathbb{2}$. The mappings of signatures to sets of sentences and to models must be functorial, while the valuation functions should be natural with respect to the signatures considered. This means that we can present an institution as a category of signatures together with a functor that associates to every signature an object of the comma category $\mathbb{S}\mathrm{et}/[|\_| \to \mathbb{2}]$.

This comma category is isomorphic to what in the literature is know as the category of *rooms* or *twisted relations* [GB92].

DEFINITION 2.2.10 (Rooms and corridors). The category $\mathbb{R}\mathrm{oom}$ of *rooms* and *corridors* is defined as follows:

The *objects* are *Boolean rooms*, that is triples $\langle S, \mathbb{M}, \models \rangle$ comprising:

- a set S of *sentences*,
- a category $\mathbb{M}$ of *models*, and
- a *satisfaction relation* $\models \subseteq |\mathbb{M}| \times S$.

The arrows $\langle S, \mathbb{M}, \models \rangle \to \langle S', \mathbb{M}', \models' \rangle$, called *corridors*, are pairs $\langle \alpha, \beta \rangle$, where

- $\alpha \colon S \to S'$ is a *sentence-translation function* and
- $\beta \colon \mathbb{M}' \to \mathbb{M}$ is a *model-reduction functor*,

such that for all $M' \in |\mathbb{M}'|$ and $\rho \in S$

$$M' \models' \alpha(\rho) \qquad \text{iff} \qquad \beta(M') \models \rho.$$

The *composition* of morphisms is defined in a componentwise manner.

Rooms are suitable for conveying the notion of local satisfaction system, associated to a fixed signature, as in studies on generalized institutions [GB85], on heterogeneous logical systems[Mos02], and on substitution systems [ŢF17].

REMARK 2.2.11. We can identify every institution $\mathcal{I}$ having the category of signatures $\mathbb{S}\mathrm{ig}$ with the functor $\mathcal{I}\colon \mathbb{S}\mathrm{ig} \to \mathbb{R}\mathrm{oom}$ that maps:

- every signature Σ to the room $\langle \mathrm{Sen}^{\mathcal{I}}(\Sigma), \mathrm{Mod}^{\mathcal{I}}(\Sigma), \models^{\mathcal{I}}_{\Sigma} \rangle$ and
- every signature morphism φ to $\langle \mathrm{Sen}^{\mathcal{I}}, \mathrm{Mod}^{\mathcal{I}} \rangle$.

As a converse, every functor $\mathcal{I}\colon \mathbb{S}\mathrm{ig} \to \mathbb{R}\mathrm{oom}$ defines an institution whose sentence functor, model functor, and satisfaction relations are given by the projections to $\mathbb{S}\mathrm{et}$ and $\mathbb{C}\mathrm{at}$ of the rooms that $\mathcal{I}$ assigns to the signatures in $\mathbb{S}\mathrm{ig}$. The satisfaction condition is automatically ensured by the fact that signature morphisms are mapped under $\mathcal{I}$ to corridors.

The equivalence between functors into $\mathbb{R}\mathrm{oom}$ and institutions can be extended to institution comorphisms. Every comorphism $\langle \Phi, \alpha, \beta \rangle\colon \mathcal{I} \to \mathcal{I}'$ can be seen as an arrow $\langle \Phi, \tau \rangle$ between the functors $\mathcal{I}\colon \mathbb{S}\mathrm{ig} \to \mathbb{R}\mathrm{oom}$ and $\mathcal{I}'\colon \mathbb{S}\mathrm{ig}' \to \mathbb{R}\mathrm{oom}$, where τ is the natural transformation $\tau\colon \mathcal{I} \Rightarrow (\Phi \mathbin{;} \mathcal{I}')$ with τ_{Σ} given by the corridor $\langle \alpha_{\Sigma}, \beta_{\Sigma} \rangle$ for every $\mathcal{I}$-signature Σ.

$$\begin{tikzcd}
\mathbb{S}\mathrm{ig} \arrow[r, "\mathcal{I}"] \arrow[d, "\Phi"'] & \mathbb{R}\mathrm{oom} \arrow[dl, phantom, ""{name=R}] \\
\mathbb{S}\mathrm{ig}' \arrow[ur, "\mathcal{I}'"'] &
\arrow[from=1-2, to=2-1, Rightarrow, shorten <=6mm, shorten >=6mm, "\tau"]
\end{tikzcd}$$

FACT 2.2.12. The category co$\mathbb{I}$ns of institution comorphisms is isomorphic with the Grothendieck category $[\_ \to \mathbb{R}\mathrm{oom}]^{\#}$.

2.3 RESIDUATED LATTICES

The class of substructural logics includes classical propositional logic, intuitionistic logic, linear logic, many-valued logics, fuzzy logics, and other notable non-classical logical systems. These were originally presented as logics that lack some of the three main structural rules of Gentzen-style sequent systems: contraction, weakening and exchange. For example, many-valued logics, fuzzy logics, and linear logic lack the contraction rule, while linear logic lacks the weakening rule [Ono03].

What brings these logics together is a shared feature, called the *residuation property*, that becomes apparent with the following equivalence presented in the context of sequent-systems formalizations [Gal+07]. In a fixed substructural logic, a formula ρ follows from formulas φ and ψ if and only if the implication $\varphi \to \rho$ follows from ψ. The derivation relation of each logic is usually captured by a sequent separator $\vdash$. Thus, we have that $\varphi, \psi \vdash \rho$ is provable if and only if so is $\psi \vdash (\varphi \to \rho)$. By replacing the comma symbol "," with a logical connective $*$, the equivalence above becomes: $\varphi * \psi \vdash \rho$ is provable if and only if so is $\psi \vdash (\varphi \to \rho)$ [KO]. In the algebraic models of the considered logic, this relation is expressed as $x * y \leq z$ iff $y \leq x \to z$, and is referred to as the *residuation law of residuated algebraic structures* [Gal+07]. Substructural logics can thus be understood as logics whose algebraic counterparts are residuated structures; because of this, implication becomes a primary connective of the language. By adding disjunction and conjunction to these structures, we obtain the lattice-ordered algebraic structures of residuated lattices.

Residuated lattices encompass basic logic algebras, Heyting algebras, MV-algebras, Boolean algebras, and lattice-ordered groups. They were first studied in the 1920s and 1930s by Krull [Kru24], Dilworth [Dil39], Ward and Dilworth [WD39], and then, four decades later, by Pavelka [Pav79b] and Balbes and Dwinger [BD74]. Following Idziak's study [Idz84], an entire new field of research flourished. Residuated lattices are known under many names, from BCK-lattices in [Idz84] and FL_{ew}-algebras in [OK85] to integral, residuated, commutative l-monoids in [Höh95]. For more details on residuated lattices, the reader is referred to the monograph [Gal+07], which started as a concise report by Kowalski and Ono [KO], but also to the surveys of Jipsen and Tsinakis [JT02], and Galatos and Jipsen [GJ09].

DEFINITION 2.3.1 (Residuated lattices). $\mathcal{L} = \langle L, \leq, \vee, \wedge, *, \to, 0, 1\rangle$ is a *residuated lattice* if $\langle L, \leq, \vee, \wedge, 0, 1\rangle$ is a bounded lattice (with supremum or join $\vee$, infimum or meet $\wedge$, smallest element 0 and greatest element 1) equipped with a monoidal structure (a commutative and associative binary operation $*$ having 1 as identity) and a binary operation $\to$ such that the *residuation law* holds: for all elements $x, y, z \in L$, $y \leq (x \to z)$ iff $x * y \leq z$.

Note that for every $x \in L$, the multiplication-with-x operation $(x * \_)$ is a functor on the partial order $\langle L, \leq\rangle$, since $(x * y) \leq (x * z)$ if $y \leq z$ (see [Ono03]). A similar observation can be made for $x \to \_$; the residuation law means that $x \to \_$ is its right adjoint. The *residuum* operation or *residual* $\to$ can be seen as a generalization of the Boolean implication, the ordinary implication arising in the particular case of the Boolean algebra $\mathbb{2}$, where $*$ coincides with the conjunction; it can be defined in a canonical way from multiplication: for every $x, y \in L$, $x \to y = \sup\{z \in L \mid x * z \leq y\}$.

We list some basic properties of residuated lattices that will be used in future proofs.

FACT 2.3.2. [KO] Let $\mathcal{L}$ be a residuated lattice and $x, y, z \in L$. Then:

1. $0 * x = 0$
2. $1 \to x = x$
3. $x \leq y$ iff $x \to y = 1$
4. $x * y \leq x$, and thus $x * y \leq x \wedge y$
5. if $x \leq y$, then $(x * z) \leq (y * z)$, $(z \to x) \leq (z \to y)$, and $(y \to z) \leq (x \to z)$
6. $x * (y \vee z) = (x * y) \vee (x * z)$

REMARK 2.3.3. We define the *negation* $\neg$ as the derived unary operation $\neg x = x \to 0$. The following equations hold in any residuated lattice:

1. $\neg(x \vee y) = \neg x \wedge \neg y$
2. $x \leq \neg\neg x$
3. $\neg\neg\neg x = \neg x$

PRESENTATION OF RESIDUATED LATTICES. In the following chapters, we often refer to the first-order presentation of residuated lattices in Figure 2.1. This is written in a CASL-like syntax [Mos04]), where sorts, operations and predicates are introduced with the keywords **sort**, **ops**, **pred**, and where we can specify operations as being commutative or associative by using the attributes comm and assoc, respectively; moreover, some operations may be specified to have units, using the unit attribute or to be idempotent, using the attribute idem. All the sentences are universally quantified over the (free) variables they contain. Note that, in this specification, some sentences are atomic, while others are implications (denoted ρ_1 if ρ_2) or equivalences (denoted ρ_1 iff ρ_2).

The residuated lattices thus specified are sometimes called *commutative* (because the monoid $\langle L, *, 1\rangle$ is commutative), *integral* (because the unit of the monoid is a greatest element of the lattice, with $x \leq 1$ for all $x \in L$), and *zero-bounded* (because there is a lowest element $0 \leq x$ for all $x \in L$).

DEFINITION 2.3.4 (Category of residuated lattices). A *morphism of residuated lattices* $\ell \colon \mathcal{L} \to \mathcal{L}'$ is a function $\ell \colon L \to L'$ that is simultaneously a morphism of lattices and of commutative monoids, and is compatible with the residuum

```
spec ResiduatedLattices =
     sort   L
     ops    0 : L, 1 : L
            _ ∨ _ : L × L ⟶ L    [comm, assoc, unit 0, idem]
            _ ∧ _ : L × L ⟶ L    [comm, assoc, unit 1, idem]
            _ ∗ _ : L × L ⟶ L    [comm, assoc, unit 1]
            _ → _ : L × L ⟶ L
     pred   _ ≤ _ : L × L

     ∀ a, b, c : L                     • a ∨ (a ∧ b) = a
     • a ∧ (a ∨ b) = a                 • a ≤ b iff a ∨ b = b
     • (a ∗ b) ≤ (a ∗ c) if b ≤ c      • b ≤ (a → c) iff (a ∗ b) ≤ c
```

FIGURE 2.1. The presentation $\langle \Sigma_{\mathbb{L}}, E_{\mathbb{L}} \rangle$ of residuated lattices

$\rightarrow$. Obviously, morphisms of residuated lattices compose (as ordinary functions do), and their composition is associative and has identities (given by identity functions). Hence, residuated lattices and their morphisms form a category, which we denote $\mathbb{L}$.

REMARK 2.3.5. The category of residuated lattices coincides with the category $\mathrm{Mod}^{\mathcal{FOL}-\mathrm{pres}}(\Sigma_{\mathbb{L}}, E_{\mathbb{L}})$ of models of the presentation $\langle \Sigma_{\mathbb{L}}, E_{\mathbb{L}} \rangle$. Moreover, residuated lattices can be even specified using universally-quantified atomic sentences [Idz84], and hence, according to Birkhoff's variety theorem, they form a variety (they are closed under homomorphic images, subalgebras, and direct products) in the category of $\Sigma_{\mathbb{L}}$-models. In particular, the residuation condition can be equationally axiomatized with the following identities [Idz84]:

- $(x * y) \rightarrow z = x \rightarrow (y \rightarrow z)$
- $(x * (x \rightarrow y)) \vee y = y$
- $(x \wedge y) \rightarrow y = 1$

By imposing additional axioms to the class of residuated lattices, we obtain several subvarieties of interest, such as the ones presented in the inclusion diagram in Figure 2.2. The axioms needed to obtain these subvarieties of non-trivial residuated lattices (with elements $0 \neq 1$) are as follows (once again, all free variables are implicitly universally quantified):

DIVISIBILITY: $x \wedge y = x * (x \rightarrow y)$,

PRELINEARITY/REPRESENTABILITY: $(x \rightarrow y) \vee (y \rightarrow x) = 1$,

LINEARITY: $x \leq y$ or $y \leq x$,

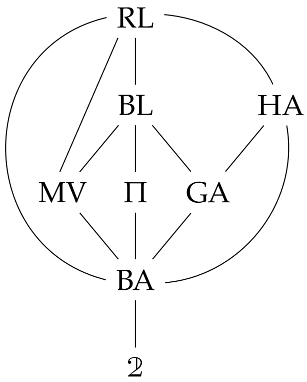


FIGURE 2.2. Subvarieties of residuated lattices ordered by inclusion

IDEMPOTENCY: $x * x = x,$

INVOLUTIVENESS: $\neg\neg x = x.$

The subvarieties of residuated lattices listed in Figure 2.2 can be characterized as follows [GJ09]:

BL: *basic logic algebras*
the algebraic counterpart of Hajek's basic logic
= divisible and prelinear residuated lattices

HA: *Heyting algebras*
the algebraic counterpart of intuitionistic logic
= residuated lattices where $*$ and $\wedge$ coincide

MV: *MV-algebras* or *Łukasiewicz algebras*
algebraic counterpart of Lukasiewicz many-valued logics
= involutive BL-algebras
= prelinear residuated lattices satisfying $x \vee y = (x \to y) \to y$

Π: *product algebras*
algebraic counterpart of product logic
= BL-algebras satisfying $\neg\neg x \leq (x \to x * y) \to y * (\neg\neg y)$

GA: *Gödel algebras*
algebraic counterpart of Gödel-Dummett logic (smallest superintuitionistic logic that is also a fuzzy logic)
= idempotent BL-algebras
= prelinear Heyting algebras

BA: *Boolean algebras*
= involutive Heyting algebras
= idempotent MV-algebras

We list below a few representative examples of residuated lattices belonging to these varieties.

EXAMPLE 2.3.6 (Heyting algebras).

1. We denote by $\mathcal{HA}_{n+1}$ the Heyting algebra with $n+1$ values: $\langle HA_{n+1}, \max, \min, \min, \rightarrow, 0, 1\rangle$, where $HA_{n+1} = \{0, 1/n, 2/n, \ldots, (n-1)/n, 1\}$, join is maximum, meet and multiplication are minimum, and $\rightarrow$ is defined as $x \rightarrow y = \max\{z \in HA \mid \min\{x, z\} \leq y\}$.
2. Similarly, $\mathcal{HA}_{\omega}$ is the Heything algebra $\langle HA_{\omega}, \max, \min, \min, \rightarrow, 0, 1\rangle$ with the underlying infinite partial order $HA_{\omega} = \{0\} \cup \{1/n \mid n \in \omega\}$, and $\mathcal{HA}_{[0,1]}$ the algebra $\langle HA_{[0,1]}, \max, \min, \min, \rightarrow, 0, 1\rangle$ with the interval $[0, 1]$ as the underlying set, and operations defined as above.

Note that these examples are in fact Gödel algebras, having the residual operation defined as

$$x \rightarrow y = \begin{cases} 1, & x \leq y \\ y, & x > y. \end{cases}$$

EXAMPLE 2.3.7 (The standard product algebra).
$\mathcal{PI}_{[0,1]} = \langle PI_{[0,1]}, \max, \min, \cdot, \rightarrow, 0, 1\rangle$ is the product algebra with $PI_{[0,1]} = [0, 1]$, multiplication defined as the ordinary product on real numbers and the residual defined as

$$x \rightarrow y = \begin{cases} 1, & x \leq y \\ y/x, & x > y. \end{cases}$$

EXAMPLE 2.3.8 (Łukasiewicz algebras).

1. $\mathcal{L}_{n+1}$ is the MV-algebra with $n+1$ values: $\langle L_{n+1}, \max, \min, *, \rightarrow, 0, 1\rangle$, where $L_{n+1} = \{0, 1/n, 2/n, \ldots, (n-1)/n, 1\}$, join is maximum, meet is minimum, multiplication is defined as $x * y = \max\{0, x + y - 1\}$, and $\rightarrow$ is defined as $x \rightarrow y = \min\{1, 1 - x + y\}$.
2. We denote by $\mathcal{L}_{[0,1]}$ the standard MV-algebra $\langle L_{[0,1]}, \max, \min, *, \rightarrow, 0, 1\rangle$, with the interval $[0, 1]$ as the underlying set, and the join, meet, multiplication and residual operations defined as above.

DEFINITION 2.3.9 (Complete residuated lattices). A residuated lattice is *complete* when its underlying lattice is complete, i.e., when joins and meets exist for arbitrary sets of values. Together with complete homomorphisms (morphisms that preserve arbitrary joins and meets) they form a subcategory, denoted $\mathbb{RL}$, of the category $\mathbb{L}$ of residuated lattices.

REMARK 2.3.10. Note that all the examples of residuated lattices presented so far are complete.

For the following results, we consider $\mathcal{L} = \langle L, \leq, \vee, \wedge, *, \rightarrow, 0, 1\rangle$ to be a complete commutative residuated lattice.

PROPOSITION 2.3.11. *For any two elements $x, y \in L$ such that $x \geq a$ and $x \rightarrow y \geq b$, we have $y \geq a * b$.*

PROOF. For any elements $x, y \in L$, we have $y \geq x * (x \rightarrow y)$ – see [Gal+07]. From the monotonicity of $*$, and because $x \geq a$ and $x \rightarrow y \geq b$, we know that $x * (x \rightarrow y) \geq a * b$, and thus $y \geq a * b$. □

PROPOSITION 2.3.12. *For any elements $x, y, z \in L$ such that $x \rightarrow y \geq a$ and $y \rightarrow z \geq b$, we have $x \rightarrow z \geq a * b$.*

PROOF. We know from [Gal+07] that $((x \rightarrow y) \rightarrow (x \rightarrow z)) \geq (y \rightarrow z)$, for any elements $x, y, z \in L$. Since $y \rightarrow z \geq b$, it follows that $((x \rightarrow y) \rightarrow (x \rightarrow z)) \geq b$. By applying Proposition 2.3.11 for $(x \rightarrow y)$ and $((x \rightarrow y) \rightarrow (x \rightarrow z))$, we obtain $x \rightarrow z \geq a * b$. □

PROPOSITION 2.3.13. *$\mathbb{RL}$ is finitely complete.*

PROOF. We start by recalling a basic category-theory result stating that a category has all finite limits whenever it has pullbacks and a terminal object (see, for example, [AHS09]).

For any cospan $\ell_1 \colon \mathcal{L}_1 \rightarrow \mathcal{L}$ and $\ell_2 \colon \mathcal{L}_2 \rightarrow \mathcal{L}$ of morphisms of complete residuated lattices, it is easy to see that there exists a pullback $\pi_1 \colon \mathcal{L}' \rightarrow \mathcal{L}_1$, $\pi_2 \colon \mathcal{L}' \rightarrow \mathcal{L}_2$, where $\mathcal{L}'$ is the lattice having the support $L' = \{(a, b) \mid a \in L_1, b \in L_2, \ell_1(a) = \ell_2(b)\}$, and the operations $\vee'$, $\wedge'$, $*'$, $\rightarrow'$ defined component-wise: for example, $\vee'$ is defined as $(a, b) \vee' (c, d) = (a \vee_1 c, b \vee_2 d)$, for any elements (a, b) and (c, d) in L'.

$$\begin{tikzcd}
\mathcal{L}' \arrow[r, "\pi_1"] \arrow[d, "\pi_2"'] & \mathcal{L}_1 \arrow[d, "\ell_1"] \\
\mathcal{L}_2 \arrow[r, "\ell_2"'] & \mathcal{L}
\end{tikzcd}$$

Finally, it can be easily checked that the residuated lattice with only one element 1, and with operations defined in the obvious way, is a terminal object of $\mathbb{RL}$. □

PROPOSITION 2.3.14. *The category* $\mathbb{RL}$ *admits a factorization system* $\langle \mathbb{E}_{\mathbb{RL}}, \mathbb{M}_{\mathbb{RL}} \rangle$.

PROOF. To define $\langle \mathbb{E}_{\mathbb{RL}}, \mathbb{M}_{\mathbb{RL}} \rangle$ we start from the standard factorization system for the underlying sets of residuated lattices: for any morphism $\ell \colon \mathcal{L} \to \mathcal{L}'$ of complete residuated lattices, we consider the factorization

$$\begin{tikzcd}
\mathcal{L} \arrow[rr, "\ell"] \arrow[dr, "e^{\ell}"'] & & \mathcal{L}' \\
& \ell(\mathcal{L}) \arrow[ur, "m^{\ell}"'] &
\end{tikzcd}$$

where we take e^{ℓ} to be a surjection and m^{ℓ} an injection on the underlying sets L, L' and $\ell(L)$, and we define the residuated lattice $\ell(\mathcal{L})$ as the lattice defined by the underlying set $\ell(L)$, and the restrictions to $\ell(L)$ of the components $\leq$, $*$, and $\to$ of $\mathcal{L}'$. □

3

LOGICS FOR CREATIVE SYSTEMS

In this chapter, we introduce a new concept that generalizes the notion of institution and we discuss in detail several examples of such logical systems that are suitable for specifying creative processes like music improvisation. In the first section, we define $\mathbb{RL}$-institutions and we formally compare them with similar existing extensions of institutions from the literature via a Grothendieck construction. In the second section, we define a framework for enriching institutions with a mechanism for dealing with soft-constraint satisfaction problems, and we show how they can be used for computational music composition. In the final section, we focus on a particular example of many-valued logics for specifying improvisations inspired by Anthony Braxton's graphic notations.

3.1 RESIDUATED-LATTICE INSTITUTIONS

MANY-VALUED. TRUE OR FALSE?

Dealing with creative systems means having to reason with imprecise concepts. Naturally, one would expect that approximate, uncertain reasoning is accommodated by the existing mainstream approaches to logical many-valuedness [Háj98; Got01]. However, while many-valued logics allow for an arbitrary number of truth values, thus achieving many-valuedness at the object level, they generally maintain meta-level statements binary, having crisp consequence relations [DBC13; Dia14]. We are interested in a natural generalization of logical systems that allows for different degrees of satisfaction of sentences by models as well as graded consequence relations between theories. This means, in essence, replacing the *true*/*false* structure of truth values of institutions with a more complex structure that accommodates various levels of satisfaction in between.

It is fair to say that there have been studies in the literature that distance themselves from the so-called *superficial many-valued logics*, in which the logical derivations are not necessarily either *true* or *false*, starting with [Pav79a] where graded consequence is considered implicitly, and [Cha88; Cha95] where the relevance of non-crisp consequence relations is explicitly addressed. These approaches imply a fixed signature for the entire logical system, but the basic ideas can be extended to a many-signature setting.

The subject has been of interest to the institution-theory community as well. Generalizing two-valued institutions to a multi-valued setting is hardly a recent endeavour, from the concept of *generalized* $\mathbb{V}$*-institutions* presented in [GB85] to *galleries* as introduced in [May85], and to monadic *generalized institutions* as defined in [EH10]. The most recently proposed construction of this sort is that of $\mathcal{L}$*-institutions* as defined in [Dia14], where the satisfaction of sentences by models is captured using a function $\models^{\mathfrak{I}}_{\Sigma} \colon |\mathrm{Mod}^{\mathfrak{I}}(\Sigma)| \times \mathrm{Sen}^{\mathfrak{I}}(\Sigma) \to \mathcal{L}$ whose codomain is a partial order. In order to accommodate other concepts studied in that paper, and also in this thesis (such as graded consequence), $\mathcal{L}$ needs to be more than a partial order; more precisely, it needs to be a complete residuated lattice $\mathcal{L} = \langle L, \leq, \vee, \wedge, *, \to, 0, 1 \rangle$. Intuitively, the underlying set of $\mathcal{L}$ provides the degrees of satisfaction (with 0 as dissatisfaction and 1 as total satisfaction), which are ordered according to $\leq$. The operation $*$ captures the accumulation of truth values that result from successive inferences, and $\to$ corresponds to entailment between two degrees of satisfaction.

As the name of the concept suggests, the lattice $\mathcal{L}$ is fixed for the entire institution. However, we are interested in formalizing situations in which the truth structure can change. For this reason, in what follows we focus on a concept that is more flexible than $\mathcal{L}$-institutions, permitting us to evaluate sentences built using symbols from different signatures over different lattices. To that end, we assume that every signature determines a corresponding complete residuated lattice and, accordingly, every signature morphism determines an appropriate change of the truth structure by a complete lattice homomorphism.

DEFINITION 3.1.1 ($\mathbb{RL}$-institution). An $\mathbb{RL}$-*institution* $\mathfrak{I}$ consists of:

- a category $\mathbb{S}\mathrm{ig}^{\mathfrak{I}}$, a functor $\mathrm{Sen}^{\mathfrak{I}}$, and a functor $\mathrm{Mod}^{\mathfrak{I}}$ as for an institution,
- a (contravariant) *truth-space* functor $\mathrm{RL}^{\mathfrak{I}} \colon (\mathbb{S}\mathrm{ig}^{\mathfrak{I}})^{\mathrm{op}} \to \mathbb{RL}$ giving for every signature a complete residuated lattice, and
- a multi-valued *satisfaction relation* $\models^{\mathfrak{I}}_{\Sigma} \colon |\mathrm{Mod}^{\mathfrak{I}}(\Sigma)| \times \mathrm{Sen}^{\mathfrak{I}}(\Sigma) \to \mathrm{RL}^{\mathfrak{I}}(\Sigma)$ for every signature Σ,

such that the equality

$$\mathrm{RL}^{\mathfrak{I}}(\varphi)\big(M' \models^{\mathfrak{I}}_{\Sigma'} \mathrm{Sen}^{\mathfrak{I}}(\varphi)(\rho)\big) \;=\; \big(\mathrm{Mod}^{\mathfrak{I}}(\varphi)(M') \models^{\mathfrak{I}}_{\Sigma} \rho\big)$$

holds for any signature morphism $\varphi\colon \Sigma \to \Sigma'$, Σ'-model M' and Σ-sentence ρ. The satisfaction relation extends, as expected, to a *graded consequence relation* over sets $E, E' \subseteq \mathrm{Sen}^{\mathcal{I}}(\Sigma)$:

$$E \vDash^{\mathcal{I}}_{\Sigma} E' \;=\; \bigwedge \{(M \vDash^{\mathcal{I}}_{\Sigma} E) \to (M \vDash^{\mathcal{I}}_{\Sigma} E') \mid M \in |\mathrm{Mod}^{\mathcal{I}}(\Sigma)|\}$$

where, for every Σ-model M, the implication between the two satisfaction values (of E and E') is given by the residual operation of the lattice $\mathrm{RL}^{\mathcal{I}}(\Sigma)$.

In what follows, we revisit two examples of logics from the previous chapter, namely propositional and first-order logic, and we present their many-valued variants as $\mathbb{RL}$-institutions.

THE $\mathbb{RL}$-INSTITUTION OF MANY-VALUED PROPOSITIONAL LOGIC – $\mathcal{PL}_{\mathbb{RL}}$

SIGNATURES. The *signatures* are pairs $\langle \mathcal{L}, \Sigma \rangle$ of complete residuated lattices $\mathcal{L}$ and propositional signatures Σ, and the *signature morphisms* $\langle \ell, \varphi \rangle\colon \langle \mathcal{L}, \Sigma \rangle \to \langle \mathcal{L}', \Sigma' \rangle$ consist of a morphism $\ell\colon \mathcal{L}' \to \mathcal{L}$ of residuated lattices and a propositional signature morphism $\varphi\colon \Sigma \to \Sigma'$.

SENTENCES. The *sentences* are built using the $*$ symbol (as a new connective) along with the usual logical connectives. The translation of a sentence along a signature morphism is defined analogously to its Boolean version.

MODELS. An $\langle \mathcal{L}, \Sigma \rangle$-*model* M is a valuation $M\colon \Sigma \to \mathcal{L}$, and the reduct of an $\langle \mathcal{L}', \Sigma' \rangle$-model M' along a signature morphism $\langle \ell, \varphi \rangle\colon \langle \mathcal{L}, \Sigma \rangle \to \langle \mathcal{L}', \Sigma' \rangle$ is given by the composition $\varphi \,;\, M' \,;\, \ell$.

LATTICES. The *truth-space* functor is defined simply as the forgetful functor that maps every signature $\langle \mathcal{L}, \Sigma \rangle$ to its underlying lattice $\mathcal{L}$.

SATISFACTION. The *satisfaction relation* is computed by induction on the structure of sentences, from the valuations of the propositional symbols, using the operations of the residuated lattice fixed in the signature.

THE $\mathbb{RL}$-INSTITUTION OF MANY-VALUED FIRST-ORDER LOGIC – $\mathcal{FOL}_{\mathbb{RL}}$

SIGNATURES. A *many-valued first-order signature* is a pair $\langle \mathcal{L}, \Sigma \rangle$ consisting of a residuated lattice $\mathcal{L}$ and a first-order signature $\Sigma = \langle S, F, P \rangle$. As in the case of $\mathcal{PL}_{\mathbb{RL}}$, *signature morphisms* $\langle \ell, \varphi \rangle\colon \langle \mathcal{L}, \Sigma \rangle \to \langle \mathcal{L}', \Sigma' \rangle$ are defined componentwise and consist of a morphism of residuated lattices $\ell\colon \mathcal{L}' \to \mathcal{L}$ and a first-order signature morphism $\varphi\colon \Sigma \to \Sigma'$.

SENTENCES. The *sentences* are built in a similar manner to the way in which the sentences of the classical (Boolean) first-order logic are built, using the logical connectives $\vee$, $\wedge$, $*$, $\rightarrow$, and the constants 0, 1. Similarly to $\mathcal{PL}_{\mathbb{RL}}$, these connectives and constants mimic the structure of residuated lattices and replace their Boolean counterparts.

MODELS. For every signature $\langle \mathcal{L}, \Sigma \rangle$, a *model* M interprets, as expected, every sort s as a set M_s and every operation symbol $\sigma \in F_{w \to s}$ as a function $M_\sigma \colon M_w \to M_s$; however, for every relation symbol $\pi \in P_w$ we consider instead of subsets of M_w, $\mathcal{L}$-valued relations $M_\pi \colon M_w \to \mathcal{L}$. *Homomorphisms* of $\langle \mathcal{L}, \Sigma \rangle$-models $h \colon M \to N$ are defined as for the Boolean case, with the difference that, for every relation symbol $\pi \in P_w$, we have that $M_\pi(m_1, \ldots, m_n) \leq N_\pi(h_w(m_1, \ldots, m_n))$, for $w = s_1 \cdots s_n$ and $m_i \in M_{s_i}$ for $i \in \overline{1, n}$[1].

For every signature morphism $\langle \ell, \varphi \rangle \colon \langle \mathcal{L}, \Sigma \rangle \to \langle \mathcal{L}', \Sigma' \rangle$, the *reduct* $M' \restriction_{\langle \ell, \varphi \rangle}$ of an $\langle \mathcal{L}', \Sigma' \rangle$-model M' is defined as the $\langle \mathcal{L}, \Sigma \rangle$-model given by $(M' \restriction_\varphi)_x = M'_{\varphi(x)}$ for every sort and function symbol x from Σ, and by $(M' \restriction_\varphi)_\pi = M'_{\varphi(\pi)} ; \ell$ for any relation symbol π from Σ.

TRUTH SPACE. The *truth-space* functor is once again the forgetful functor mapping every signature to its underlying lattice.

SATISFACTION. The satisfaction between models and sentences is defined inductively on the structure of sentences and is based on the valuation of terms in models. Given an $\langle \mathcal{L}, \Sigma \rangle$-model M:

- for an F-term $\sigma(\overline{t})$: $M_{\sigma(\overline{t})} = M_\sigma(M_{\overline{t}})$,
- for equational atoms: $(M \vDash_{\langle \mathcal{L}, \Sigma \rangle} t =_s t') = \begin{cases} 1, & \text{if } M_t = M_{t'} \\ 0, & \text{otherwise} \end{cases}$
- for relational atoms: $(M \vDash_{\langle \mathcal{L}, \Sigma \rangle} \pi(\overline{t})) = M_\pi(M_{\overline{t}})$,
- $(M \vDash_{\langle \mathcal{L}, \Sigma \rangle} \rho_1 \wedge \rho_2) = (M \vDash_{\langle \mathcal{L}, \Sigma \rangle} \rho_1) \wedge (M \vDash_{\langle \mathcal{L}, \Sigma \rangle} \rho_2)$, and similarly for the connectives $\vee$, $\rightarrow$ and $*$,
- $(M \vDash_{\langle \mathcal{L}, \Sigma \rangle} \exists X . \rho) = \bigvee \{ M' \vDash_{\langle \mathcal{L}, S, F \uplus X, P \rangle} \rho \mid M' \restriction_{\langle \mathcal{L}, \Sigma \rangle} = M \}$, and
- $(M \vDash_{\langle \mathcal{L}, \Sigma \rangle} \forall X . \rho) = \bigwedge \{ M' \vDash_{\langle \mathcal{L}, S, F \uplus X, P \rangle} \rho \mid M' \restriction_{\langle \mathcal{L}, \Sigma \rangle} = M \}$.

REMARK 3.1.2. The many-valued variant of first-order logic described above will be used sparingly in this thesis, primarily in theoretical examples such as Example 3.1.16. However, it should not be omitted, as we consider it to be a benchmark $\mathbb{RL}$-institution, just as the Boolean version $\mathcal{FOL}$ is considered

[1] We denote by $\overline{1, n}$ the set $\{1, \ldots, n\}$.

a standard example of institution. It has the advantage of being a rich and nuanced logical system, and so it helps develop a better understanding of $\mathbb{RL}$-institutions.

We revisit the many-valued first-order logic in the last section of this chapter, where we focus on a first-order encoding of the logic of Braxton's graphic notation for capturing improvisation scores. In that context, Example 3.3.6 shows how one could specify musical fragments in $\mathcal{FOL}_{\mathbb{RL}}$. This indicates how many-valued first-order logic could be used as a basis for the music-improvisation examples in Chapter 4, or other examples of service-oriented processes. Whilst that is possible, we choose to use other, more specialised logics, such as the institution of first-order soft-constraint satisfaction problems presented in Section 3.2 or the logic of Braxton's graphic notation from Section 3.3 for reasons pertaining to notation suitability and expressivity.

The notion of institution (co)morphism can be generalized to the many-valued case as well.

DEFINITION 3.1.3 (Comorphism of $\mathbb{RL}$-institutions). Given two $\mathbb{RL}$-institutions $\mathcal{I}$ and $\mathcal{I}'$, a *many-valued institution comorphism* $\langle \Phi, \alpha, \beta, \lambda \rangle \colon \mathcal{I} \to \mathcal{I}'$ consists of:

- a signature functor $\Phi \colon \mathbb{S}\mathrm{ig}^{\mathcal{I}} \to \mathbb{S}\mathrm{ig}^{\mathcal{I}'}$,
- a natural transformation $\alpha \colon \mathrm{Sen}^{\mathcal{I}} \Rightarrow \Phi \mathbin{;} \mathrm{Sen}^{\mathcal{I}'}$,
- a natural transformation $\beta \colon \Phi^{\mathrm{op}} \mathbin{;} \mathrm{Mod}^{\mathcal{I}'} \Rightarrow \mathrm{Mod}^{\mathcal{I}}$, and
- a natural transformation $\lambda \colon \Phi^{\mathrm{op}} \mathbin{;} \mathrm{RL}^{\mathcal{I}'} \Rightarrow \mathrm{RL}^{\mathcal{I}}$

such that the following *satisfaction condition* holds for any $\mathcal{I}$-signature Σ, $\Phi(\Sigma)$-model M', and Σ-sentence ρ:

$$\lambda_\Sigma(M' \vDash^{\mathcal{I}'}_{\Phi(\Sigma)} \alpha_\Sigma(\rho)) \;=\; (\beta_\Sigma(M') \vDash^{\mathcal{I}}_{\Sigma} \rho).$$

DEFINITION 3.1.4 (co$\mathbb{RL}\mathbb{I}$ns). The comorphisms of Definition 3.1.3 can be composed componentwise; their composition is associative and has identities, thus giving rise to the category co$\mathbb{RL}\mathbb{I}$ns of $\mathbb{RL}$-institutions and $\mathbb{RL}$-institution comorphisms.

CHANGING THE TRUTH SPACE

There is a special relationship between the $\mathcal{L}$-institutions defined in [Dia14] and $\mathbb{RL}$-institutions. On the one hand, every $\mathcal{L}$-institution can be defined as a *plain* $\mathbb{RL}$-*institution* with a constant truth-space functor: for any signature Σ, $\mathrm{RL}(\Sigma) = \mathcal{L}$; and, for every signature morphism $\varphi\colon \Sigma \to \Sigma'$, $\mathrm{RL}(\varphi) = \mathrm{id}_{\mathcal{L}}$. On the other hand, many $\mathbb{RL}$-institutions (although not all, but all that are of interest in this thesis) can be built from $\mathcal{L}$-institutions through a Grothendieck construction [Dia02; Mos02] as described later in this section.

NOTATIONS. In what follows, we generally denote by $\mathcal{I}_{\mathbb{RL}}$ an arbitrary $\mathbb{RL}$-institution, by $\mathcal{I}_{\mathcal{L}}$ an arbitrary $\mathcal{L}$-institution and, unless stated otherwise, by $\mathcal{I}$ a Boolean institution. Except when explicitly stated in the text, these notations do not imply the existence of a way to build $\mathcal{I}_{\mathbb{RL}}$ from $\mathcal{I}_{\mathcal{L}}$ or from $\mathcal{I}$.

It is possible, however, to derive an $\mathcal{L}$-institution $\mathcal{I}_{\mathcal{L}}$ from any $\mathbb{RL}$-institution $\mathcal{I}_{\mathbb{RL}}$ by restricting its category of signatures only to those signatures whose image under the truth-space functor is the lattice $\mathcal{L}$. In particular, when $\mathcal{L} = \mathbb{2}$, we obtain a Boolean institution. To present this process rigorously, it is useful to look at $\mathbb{RL}$-institutions from a functorial perspective, as in Remark 2.2.11.

MANY-VALUED INSTITUTIONS AS FUNCTORS

The first step is to extend the concept of *room* from the Boolean setting (see Definition 2.2.10) to the many-valued one, with satisfaction evaluated over a fixed residuated lattice $\mathcal{L}$.

DEFINITION 3.1.5 ($\mathcal{L}$-rooms and corridors). The category $\mathcal{L}$-$\mathbb{R}$oom has as objects triples $\langle S, \mathbb{M}, \models\rangle$, where S and $\mathbb{M}$ are as for Boolean rooms, and the satisfaction relation is many-valued, $\models\colon |\mathbb{M}| \times S \to \mathcal{L}$. The arrows, called $\mathcal{L}$-*corridors*, between $\langle S, \mathbb{M}, \models\rangle$ and $\langle S', \mathbb{M}', \models'\rangle$ are defined as pairs $\langle \alpha\colon S \to S', \beta\colon \mathbb{M} \to \mathbb{M}'\rangle$ such that the following satisfaction condition holds for all $M' \in |\mathbb{M}'|$ and $\rho \in S$:

$$(M' \models' \alpha(\rho)) \;=\; (\beta(M') \models \rho).$$

By allowing the satisfaction relations of different rooms to be evaluated over different lattices, we obtain the category of many-valued rooms over arbitrary residuated lattices.

DEFINITION 3.1.6 ($\mathbb{RL}$-rooms and corridors). The objects of $\mathbb{RL}$-$\mathbb{R}$oom are tuples $\langle S, \mathbb{M}, \mathcal{L}, \models\rangle$, where S, $\mathbb{M}$, and $\models$ form an $\mathcal{L}$-room. Corridors between $\mathbb{RL}$-rooms $\langle S, \mathbb{M}, \mathcal{L}, \models\rangle$ and $\langle S', \mathbb{M}', \mathcal{L}', \models'\rangle$ are triples $\langle \alpha, \beta, \lambda\rangle$, where $\alpha\colon S \to S'$ is a function, $\beta\colon \mathbb{M}' \to \mathbb{M}$ is a functor, and $\lambda\colon \mathcal{L}' \to \mathcal{L}$ is a morphism of complete residuated lattices such that

$$\lambda(M' \models' \alpha(\rho)) \;=\; (\beta(M') \models \rho)$$

for all $M' \in |\mathbb{M}'|$ and $\rho \in S$.

DEFINITION 3.1.7 ($+_\mathcal{L}$ functor). For every lattice $\mathcal{L} \in |\mathbb{RL}|$, we define a functor $+_\mathcal{L} \colon \mathcal{L}\text{-}\mathbb{R}\mathrm{oom} \to \mathbb{RL}\text{-}\mathbb{R}\mathrm{oom}$ that maps every $\mathcal{L}$-room $\langle S, \mathbb{M}, \models\rangle$ to the $\mathbb{RL}$-room $\langle S, \mathbb{M}, \mathcal{L}, \models\rangle$, and every $\mathcal{L}$-corridor $\langle \alpha, \beta\rangle$ to $\langle \alpha, \beta, \mathrm{id}_\mathcal{L}\rangle$. This functor is an embedding: it is full and faithful, and injective on objects.

FACT 3.1.8. Similar to the manner in which we defined co$\mathbb{I}$ns in Fact 2.2.12, we can define the comorphism-based categories of $\mathcal{L}$-institutions and $\mathbb{RL}$-institutions, as co$\mathcal{L}\mathbb{I}$ns $\simeq [\_ \to \mathcal{L}\text{-}\mathbb{R}\mathrm{oom}]^\#$ and co$\mathbb{RL}\mathbb{I}$ns $\simeq [\_ \to \mathbb{RL}\text{-}\mathbb{R}\mathrm{oom}]^\#$.

Let us study in more detail the relationship between these categories.

FROM $\mathcal{L}$-INSTITUTIONS TO $\mathbb{RL}$-INSTITUTIONS. By Fact 2.1.10, for every lattice $\mathcal{L} \in |\mathbb{RL}|$, the functor $+_\mathcal{L} \colon \mathcal{L}\text{-}\mathbb{R}\mathrm{oom} \to \mathbb{RL}\text{-}\mathbb{R}\mathrm{oom}$ induces a functor $[+_\mathcal{L}]^\# \colon [\_ \to \mathcal{L}\text{-}\mathbb{R}\mathrm{oom}]^\# \to [\_ \to \mathbb{RL}\text{-}\mathbb{R}\mathrm{oom}]^\#$, i.e. between co$\mathcal{L}\mathbb{I}$ns and co$\mathbb{RL}\mathbb{I}$ns, that formalizes the correspondence between $\mathcal{L}$-institutions and *plain* $\mathbb{RL}$-institutions that were introduced at the beginning of this section.

$$\begin{tikzcd}
\mathbb{S}\mathrm{ig} \arrow[r, "\mathcal{I}_\mathcal{L}"] \arrow[d, "\Phi"'] & \mathcal{L}\text{-}\mathbb{R}\mathrm{oom} \arrow[r, "+_\mathcal{L}"] \arrow[dl, Rightarrow, shorten=6mm, "\tau"] & \mathbb{RL}\text{-}\mathbb{R}\mathrm{oom} \\
\mathbb{S}\mathrm{ig}' \arrow[ur, "\mathcal{I}'_\mathcal{L}"'] & &
\end{tikzcd}$$

FROM $\mathbb{RL}$-INSTITUTIONS TO $\mathcal{L}$-INSTITUTIONS. For every $\mathcal{L} \in |\mathbb{RL}|$, and every $\mathbb{RL}$-institution $\mathcal{I}_\mathbb{RL}$, we can obtain an $\mathcal{L}$-institution using the preimage of the truth-space functor. We define the mapping $(\_\!\upharpoonright_\mathcal{L}) \colon |\mathrm{co}\mathbb{RL}\mathbb{I}\mathrm{ns}| \to |\mathrm{co}\mathcal{L}\mathbb{I}\mathrm{ns}|$ that associates with every $\mathbb{RL}$-institution $\mathcal{I}_\mathbb{RL}$ an $\mathcal{L}$-institution $\mathcal{I}_\mathbb{RL}\!\upharpoonright_\mathcal{L} \colon (\mathrm{RL}^{\mathcal{I}_\mathbb{RL}})^{-1}(\mathcal{L}) \to \mathcal{L}\text{-}\mathbb{R}\mathrm{oom}$ from the subcategory of signatures whose image through the truth-space functor $\mathrm{RL}^{\mathcal{I}_\mathbb{RL}}$ is the lattice $\mathcal{L}$. That $\mathcal{L}$-institution maps every signature Σ to the $\mathcal{L}$-room $\langle S, M, \models\rangle$ when $\mathcal{I}_\mathbb{RL}(\Sigma) = \langle S, M, \mathcal{L}, \models\rangle$, and every signature morphism φ to the $\mathcal{L}$-corridor $\langle \alpha, \beta\rangle$ when $\mathcal{I}_\mathbb{RL}(\varphi) = \langle \alpha, \beta, \mathrm{id}_\mathcal{L}\rangle$.

FACT 3.1.9. The functor $\mathcal{I}_\mathbb{RL}\!\upharpoonright_\mathcal{L}$ together with the inclusion of categories $(\mathrm{RL}^{\mathcal{I}_\mathbb{RL}})^{-1}(\mathcal{L}) \subseteq \mathbb{S}\mathrm{ig}^{\mathcal{I}_\mathbb{RL}}$ form a pullback of $\mathcal{I}_\mathbb{RL}$ and $+_\mathcal{L}$. Moreover, $\mathcal{I}_\mathbb{RL}\!\upharpoonright_\mathcal{L}$ is the unique functor with this property.

$$\begin{tikzcd}
(\mathrm{RL}^{\mathcal{I}_{\mathbb{RL}}})^{-1}(\mathcal{L}) \arrow[r, "\subseteq"] \arrow[d, "\mathcal{I}_{\mathbb{RL}}\restriction_{\mathcal{L}}"'] & \mathbb{S}\mathrm{ig}^{\mathcal{I}_{\mathbb{RL}}} \arrow[d, "\mathcal{I}_{\mathbb{RL}}"] \\
\mathcal{L}\text{-}\mathbb{R}\mathrm{oom} \arrow[r, "+_{\mathcal{L}}"'] & \mathbb{RL}\text{-}\mathbb{R}\mathrm{oom}
\end{tikzcd}$$

Note that the mapping $\_\restriction_{\mathcal{L}}$ cannot be extended on the arrows of $\mathrm{co}\mathbb{RL}\mathbb{I}\mathrm{ns}$, because the signature-functor component of $\mathbb{RL}$-institution comorphisms does not necessarily preserve the underlying lattices of signatures. This is also the reason why we cannot define the category $\mathrm{co}\mathbb{RL}\mathbb{I}\mathrm{ns}$ simply as a Grothendieck construction over the categories $\mathrm{co}\mathcal{L}\mathbb{I}\mathrm{ns}$ of $\mathcal{L}$-institutions. What we can do, using the Grothendieck construction, is to build particular $\mathbb{RL}$-institutions (such as $\mathcal{PL}_{\mathbb{RL}}$ and $\mathcal{FOL}_{\mathbb{RL}}$) out of indexed $\mathcal{L}$-institutions.

GROTHENDIECK MANY-VALUED INSTITUTIONS

Two kinds of ingredients are necessary for obtaining a many-valued $\mathbb{RL}$-institution via a Grothendieck construction:

1. for every complete residuated lattice $\mathcal{L}$, an $\mathcal{L}$-institution $\mathrm{I}(\mathcal{L})$, which we regard as a plain $\mathbb{RL}$-institution, and
2. for every morphism $\ell\colon \mathcal{L}' \to \mathcal{L}$ of complete residuated lattices, an encoding $\mathrm{I}(\ell)$ of $\mathrm{I}(\mathcal{L})$ into $\mathrm{I}(\mathcal{L}')$, formalized as a comorphism of $\mathbb{RL}$-institutions.

EXAMPLE 3.1.10. [Between many-valued propositional logics] Let $\mathrm{PL}(\mathcal{L})$ and $\mathrm{PL}(\mathcal{L}')$ be the $\mathcal{L}$- and $\mathcal{L}'$-institution of many-valued propositional logic – defined like the Boolean version of propositional logic, but with sentences evaluated over $\mathcal{L}$ and $\mathcal{L}'$, respectively. Every morphism of complete residuated lattices $\ell\colon \mathcal{L}' \to \mathcal{L}$ determines an $\mathbb{RL}$-comorphism $\mathrm{PL}(\ell) = (\Phi, \alpha, \beta, \lambda)\colon \mathrm{PL}(\mathcal{L}) \to \mathrm{PL}(\mathcal{L}')$ that consists of:[2]

- the identity functor $\Phi = \mathrm{id}_{\mathrm{Sig}^{\mathcal{PL}}}$ of propositional signatures,
- the natural transformation α whose components are all identities,
- the natural transformation β given by $\beta_\Sigma(M) = M\,;\ell$ for any Σ-model M,
- the natural transformation λ with components $\lambda_\Sigma = \ell$.

It is easy to check that the satisfaction condition holds, because the equalities

$$\lambda_\Sigma(M' \vDash_{\Phi(\Sigma)} \alpha_\Sigma(\rho)) = \ell(M' \vDash_{\Phi(\Sigma)} \rho) = \ell(M'(\rho)) = (M'\,;\ell)(\rho) = \beta_\Sigma(M') \vDash_\Sigma \rho$$

[2] Notice that the comorphism is contravariant with respect to the direction of ℓ. This is justified by the way in which the model reducts are defined.

hold for any signature Σ, $\Phi(\Sigma)$-model M', and atomic Σ-sentence ρ.

In other words, we regard every $\mathcal{L}$-institution of propositional logic as a plain $\mathbb{RL}$-institution $\mathrm{PL}(\mathcal{L})$ and we consider, for every morphism of residuated lattices ℓ, a corresponding comorphism $\mathrm{PL}(\ell)$ of $\mathbb{RL}$-institutions. We then flatten the class of plain $\mathbb{RL}$-institutions thus obtained into a single institution, denoted $\mathrm{PL}^{\#}$: its category of signatures has as objects pairs of residuated lattices and propositional-logic signatures, and as arrows pairs of morphisms of residuated lattices and propositional-logic signature morphisms. The sentences and models of a signature $\langle \mathcal{L}, \Sigma \rangle$ are the propositional sentences and models corresponding to Σ in the $\mathcal{L}$-institution of propositional logic, while the truth-space functor is defined as the projection on the lattice component. Similarly, the satisfaction of an $\langle \mathcal{L}, \Sigma \rangle$-sentence ρ by a model M is defined as the satisfaction of the Σ-sentence ρ by M in the $\mathcal{L}$-institution $\mathrm{PL}(\mathcal{L})$.

In a more general setting, for presenting $\mathbb{RL}$-institutions as the result of flattening many-valued Grothendieck institutions, we follow closely [Mos02]. We discuss this in detail in the remaining part of this section.

DEFINITION 3.1.11 (Indexed many-valued institution). Given a category $\mathbb{I}$ of indices, an $\mathbb{I}$-*indexed* $\mathbb{RL}$-*institution* is a functor $\mathrm{I}\colon \mathbb{I}^{\mathrm{op}} \to \mathrm{co}\mathbb{RL}\mathbb{I}\mathrm{ns}$ into the category of comorphisms of $\mathbb{RL}$-institutions.

For an indexed $\mathbb{RL}$-institution I, we use the notation $\langle \Phi^u, \tau^u \rangle$ for the comorphism $\mathrm{I}(u)$, where τ^u_Σ is given by the corridor $\langle \alpha^u_\Sigma, \beta^u_\Sigma, \lambda^u_\Sigma \rangle$.

DEFINITION 3.1.12 (Grothendieck many-valued institution). Given an indexed $\mathbb{RL}$-institution $\mathrm{I}\colon \mathbb{I}^{\mathrm{op}} \to \mathrm{co}\mathbb{RL}\mathbb{I}\mathrm{ns}$, we define the *Grothendieck* $\mathbb{RL}$-*institution* $\mathrm{I}^{\#}$ as follows:

- the signatures are pairs $\langle i, \Sigma \rangle$, where $i \in |\mathbb{I}|$ and Σ is a signature in $\mathrm{I}(i)$,
- signature morphisms $\langle u, \varphi \rangle\colon \langle i, \Sigma \rangle \to \langle i', \Sigma' \rangle$ are pairs of morphisms $u\colon i' \to i$ in $\mathbb{I}$ and signature morphisms $\varphi\colon \Phi^u(\Sigma) \to \Sigma'$ in $\mathbb{I}(i')$,
- the composition of signature morphisms is given by $\langle u, \varphi \rangle \mathbin{;} \langle u', \varphi' \rangle = \langle u' \mathbin{;} u, \Phi^{u'}(\varphi) \mathbin{;} \varphi' \rangle$,
- $\mathrm{I}^{\#}(i, \Sigma) = \mathrm{I}(i)(\Sigma)$, and
- $\mathrm{I}^{\#}(u, \varphi) = \mathrm{I}(i)(\Sigma) \xrightarrow{\tau^u_\Sigma} \mathrm{I}(i')(\Phi^u(\Sigma)) \xrightarrow{\mathrm{I}(i')(\varphi)} \mathrm{I}(i')(\Sigma')$.

REMARK 3.1.13. Even though the Grothendieck construction above differs from the standard one presented in Section 2.1, it can still be obtained from the standard construction through dualization. What the two constructions have in common, and what legitimates bearing the name Grothendieck construction, is the fact that $\mathrm{I}^{\#}$ is the vertex of a lax colimit (in $\mathrm{co}\mathbb{RLI}\mathrm{ns}$ instead of $\mathbb{C}\mathrm{at}$).

FACT 3.1.14. The category $\mathrm{co}\mathbb{RLI}\mathrm{ns}$ admits a Grothendieck construction for every indexed many-valued institution $\mathrm{I}\colon \mathbb{I}^{\mathrm{op}} \to \mathrm{co}\mathbb{RLI}\mathrm{ns}$. We denote by $\mathrm{I}^{\#}$ the vertex of the lax colimit $\mu\colon \mathrm{I} \to \mathrm{I}^{\#}$ of I in $\mathrm{co}\mathbb{RLI}\mathrm{ns}$.

We describe in what follows how to present concretely an $\mathbb{RL}$-institution $\mathcal{I}_{\mathbb{RL}}$ (belonging to a particular class of many-valued logics) as a Grothendieck many-valued institution. We assume that there exists a category $\mathbb{S}\mathrm{ig}_0$ such that $\mathbb{S}\mathrm{ig}^{\mathcal{I}_{\mathbb{RL}}} = \mathbb{RL}^{\mathrm{op}} \times \mathbb{S}\mathrm{ig}_0$, i.e. $\mathcal{I}_{\mathbb{RL}}$ has as signatures pairs $\langle \mathcal{L}, \Sigma \rangle$ of complete residuated lattices $\mathcal{L}$ and objects (signatures) Σ in $\mathbb{S}\mathrm{ig}_0$. In addition, we assume that the functor $\mathrm{RL}^{\mathcal{I}_{\mathbb{RL}}}$ from $(\mathbb{RL}^{\mathrm{op}} \times \mathbb{S}\mathrm{ig}_0)^{\mathrm{op}} = \mathbb{RL} \times \mathbb{S}\mathrm{ig}_0^{\mathrm{op}}$ to $\mathbb{RL}$ is the contravariant projection on the residuated-lattice component.

This allows us to define a corresponding indexed many-valued institution $\mathrm{I}\colon \mathbb{RL}^{\mathrm{op}} \to \mathrm{co}\mathbb{RLI}\mathrm{ns}$ for $\mathcal{I}_{\mathbb{RL}}$ using the category of complete residuated lattices as the category of indices. To that end, we denote by $\mathcal{I}_{\mathcal{L}}$ the $\mathcal{L}$-restriction of $\mathcal{I}_{\mathbb{RL}}$, i.e. $\mathcal{I}_{\mathcal{L}} = \mathcal{I}_{\mathbb{RL}} \restriction_{\mathcal{L}}$ as in Fact 3.1.9, and by $\iota_{\mathcal{L}}$ the embedding of $\mathbb{S}\mathrm{ig}_0$ into $\mathbb{S}\mathrm{ig}^{\mathcal{I}_{\mathcal{L}}}$ that maps each signature Σ to the pair $\langle \mathcal{L}, \Sigma \rangle$:

- for every lattice $\mathcal{L} \in |\mathbb{RL}|$, $\mathrm{I}(\mathcal{L}) = (\iota_{\mathcal{L}} \mathbin{;} \mathcal{I}_{\mathcal{L}} \mathbin{;} +_{\mathcal{L}})\colon \mathbb{S}\mathrm{ig}_0 \to \mathbb{RL}\text{-}\mathbb{R}\mathrm{oom}$.
- for every morphism $\ell\colon \mathcal{L}' \to \mathcal{L} \in \mathbb{RL}$, $\mathrm{I}(\ell) = \langle \Phi^{\ell}, \alpha^{\ell}, \beta^{\ell}, \lambda^{\ell} \rangle\colon \mathrm{I}(\mathcal{L}) \to \mathrm{I}(\mathcal{L}')$, where Φ^{ℓ} is the identity of $\mathbb{S}\mathrm{ig}_0$, and for every signature $\Sigma \in |\mathbb{S}\mathrm{ig}_0|$, α^{ℓ}_{Σ}, β^{ℓ}_{Σ}, and λ^{ℓ}_{Σ} are the components of the $\mathbb{RL}$-corridor $\mathcal{I}_{\mathbb{RL}}(\ell, \mathrm{id}_{\Sigma})$.

Note that the satisfaction condition of the comorphism $\mathrm{I}(\ell)$

$$\beta^{\ell}_{\Sigma}(M') \vDash^{\mathrm{I}(\mathcal{L})}_{\Sigma} \rho \ = \ \lambda^{\ell}_{\Sigma}(M' \vDash^{\mathrm{I}(\mathcal{L}')}_{\Sigma} \alpha^{\ell}_{\Sigma}(\rho))$$

which can be rewritten as

$$M' \restriction_{\langle \ell, \mathrm{id}_{\Sigma} \rangle} \vDash^{\mathcal{I}_{\mathbb{RL}}}_{\langle \mathcal{L}, \Sigma \rangle} \rho \ = \ \mathrm{RL}^{\mathcal{I}_{\mathbb{RL}}}(\ell, \mathrm{id}_{\Sigma})(M' \vDash^{\mathcal{I}_{\mathbb{RL}}}_{\langle \mathcal{L}', \Sigma \rangle} \langle \ell, \mathrm{id}_{\Sigma} \rangle(\rho))$$

is guaranteed to hold by the satisfaction condition for the signature morphism $\langle \ell, \mathrm{id}_{\Sigma} \rangle$ in the $\mathbb{RL}$-institution $\mathcal{I}_{\mathbb{RL}}$.

PROPOSITION 3.1.15. *By flattening the indexed many-valued institution defined as above, we get* $\mathrm{I}^{\#} = \mathcal{I}_{\mathbb{RL}}$.

PROOF. We start from the definition of $\mathcal{I}_{\mathbb{RL}}$ as a functor into $\mathbb{RL}$-$\mathbb{R}\mathrm{oom}$, and show that $\mathcal{I}_{\mathbb{RL}}$ and $\mathrm{I}^{\#}$ coincide on both objects and morphisms.

OBJECTS. Considering a lattice $\mathcal{L} \in |\mathbb{RL}|$ and $\Sigma \in |\mathbb{S}\mathrm{ig}_0|$,

$$\mathrm{I}^{\#}(\mathcal{L}, \Sigma) = \mathrm{I}(\mathcal{L})(\Sigma) = (\iota_{\mathcal{L}} \,;\, \mathfrak{I}_{\mathcal{L}} \,;\, +_{\mathcal{L}})(\Sigma) = \mathfrak{I}_{\mathbb{RL}}(\mathcal{L}, \Sigma).$$

MORPHISMS. For a signature morphism $\langle \ell, \varphi \rangle \colon \langle \mathcal{L}, \Sigma \rangle \to \langle \mathcal{L}', \Sigma' \rangle$,

$$\begin{aligned} \mathrm{I}^{\#}(\ell, \varphi) &= \langle \alpha^{\ell}_{\Sigma}, \beta^{\ell}_{\Sigma}, \lambda^{\ell}_{\Sigma} \rangle \,;\, \mathrm{I}(\mathcal{L}')(\varphi) \\ &= \langle \alpha^{\ell}_{\Sigma}, \beta^{\ell}_{\Sigma}, \lambda^{\ell}_{\Sigma} \rangle \,;\, (\iota_{\mathcal{L}'} \,;\, \mathfrak{I}_{\mathcal{L}'} \,;\, +_{\mathcal{L}'})(\varphi) \\ &= \mathfrak{I}_{\mathbb{RL}}(\ell, \mathrm{id}_{\Sigma}) \,;\, \mathfrak{I}_{\mathbb{RL}}(\mathrm{id}_{\mathcal{L}'}, \varphi) \\ &= \mathfrak{I}_{\mathbb{RL}}(\mathrm{id}_{\mathcal{L}'} \,;\, \ell, \mathrm{id}_{\Sigma} \,;\, \varphi) = \mathfrak{I}_{\mathbb{RL}}(\ell, \varphi) \end{aligned}$$

□

EXAMPLE 3.1.16 (Many-valued first-order logic as a Grothendieck institution). Consider the indexed many-valued institution $\mathrm{FOL} \colon \mathbb{RL}^{\mathrm{op}} \to \mathrm{co}\mathbb{RL}\mathbb{I}\mathrm{ns}$ such that for every lattice $\mathcal{L}$, $\mathrm{FOL}(\mathcal{L})$ corresponds to the $\mathcal{L}$-institution of first-order logic, and for every morphism $\ell \colon \mathcal{L}' \to \mathcal{L}$, $\mathrm{FOL}(\ell)$ is the comorphism that leaves signatures and sentences unchanged, and post-composes the interpretation of predicates with ℓ. Notice that this fits within the premises of Proposition 3.1.15; therefore, by flattening, we can obtain the $\mathbb{RL}$-institution $\mathcal{FOL}_{\mathbb{RL}}$ as $\mathrm{FOL}^{\#}$.

3.2 LOGICS FOR CONSTRAINT COMPOSITION

Constraint programming has proved in the past decades to be appropriate for computational music composition and modelling music-theory disciplines such as harmony, rhythm, instrumentation and counterpoint [AM11]. The declarative nature and the modularity of such *constraint satisfaction systems* match the way in which composition rules are commonly expressed in standard music theory.

Deciding the satisfaction of certain properties or rules based on the *true/false* dichotomy is however often inadequate for the purpose of composing music, even more so when dealing with improvisation. In this section we show that a many-valued approach mitigates the problem of expressing loose rules and provides more flexibility in writing musical guidelines.

To achieve this, we investigate the generalization of constraint satisfaction problems (CSP) to a many-valued setting, known as soft-constraint satisfaction problems [Bis+99]. The goal of relaxing the constraint-based frameworks for music composition seems to have been pursued for at least a decade in the community of constraint music composition, although no solutions have been implemented so far. A notable exception is the tool *Constraint-Muse* [Höl+09], which is the prototype of a system based on soft constraints for music therapy.

In a nutshell, our aim is to enrich specifications of *musical spaces* with soft constraints in order to capture music-composition and improvisation processes. In this section, we extend the institution-based theory of algebraic specifications to address soft CSP. Such an extension is essential for providing a logic-independent foundation that, on the one hand, can be used to support different specification languages and, on the other hand, can be integrated in development environments that offer automated theorem-proving support for the specification and analysis of systems – for example, *The Heterogeneous Tool Set* [MML07]. In this way, we bring new levels of flexibility to constraint music composition: firstly, we adapt constraint music programming to a multi-valued setting; moreover, we increase the expressivity of these composition systems by allowing the constraints to be captured as sentences in a diverse range of logical systems, not only as variables in constraint systems that are based, essentially, on propositional logic. This allows us to develop further mechanisms for reasoning about composition processes, and thus exceed the scope of the existing formalizations of music programming as constraint satisfaction problems, which focus only on obtaining a solution, and not on modelling the composition process itself.

3.2.1 SOFT-CONSTRAINT INSTITUTIONS

SOFT-CONSTRAINT SATISFACTION PROBLEMS

Before dealing with the logics for constraint composition, a short introduction to soft-constraints satisfaction problems is in order. Several alternative frameworks have been developed for dealing with soft constraints; the main two approaches are the frameworks of SCSP [BMR97] and VCSP [SFV95; Coh+03], which generalise the classical crisp variant of CSP by evaluating constraints over c-semirings and valuation structures, respectively. For the purpose of this thesis, we focus on the SCSP framework, following closely the presentation in [BMR97]; VCSP could be presented, however, in a similar way (see Remark 3.2.10).

DEFINITION 3.2.1 (C-semiring). A *c-semiring* $\mathcal{S}$ is a tuple $\langle S, +, \times, 0, 1 \rangle$, where

- S is a set with two distinguished elements, 0 and 1;
- $+$ is a commutative, associative, idempotent operation over potentially infinite subsets of S with unit element 0. When the sum is applied to a set of

two elements, we use the symbol +; in general the symbol $\sum$ is used, with the following properties:

  - $\sum(\{a\}) = a$ for every $a \in S$,
  - $\sum(\emptyset) = 0$, $\sum(S) = 1$,
  - $\sum(\cup\{S_i \mid i \in J\}) = \sum(\{\sum(S_i) \mid i \in J\})$, for every set of indices J;

- $\times$ is a binary, commutative, associative operation with unit 1, for which 0 is absorbing, meaning that $a \times 0 = 0$ for every $a \in S$;
- $\times$ distributes over the sum: $a \times \sum(\{b_i \mid i \in J\}) = \sum(\{a \times b_i \mid i \in J\})$.

FACT 3.2.2. For every c-semiring $\mathcal{S}$, we can define a partial order $\leq$ based on the additive operation: for every $a, b \in S$, $a \leq b$ if and only if $a + b = b$. If the operation $\times$ is idempotent, the structure $\mathcal{S} = \langle S, \leq, +, \times, 0, 1\rangle$ defined by the c-semiring together with the partial order is a complete distributive lattice.

Intuitively, the underlying set of a c-semiring represents a space of degrees of satisfaction of constraints, with 0 representing dissatisfaction and 1 representing total satisfaction, while the operations + and $\times$ are used for choice and composition. A *constraint system* fixes a constraint-satisfaction semiring structure, a global set of variables and their domain.

DEFINITION 3.2.3 (Constraint system). A *constraint system* CS is a tuple $\langle \mathcal{S}, V, D\rangle$ consisting of:

- a c-semiring $\mathcal{S}$,
- a finite set V of *variables*, and
- a finite set D representing the *domain* of the variables.

A *constraint* over a constraint system specifies a subset of variables to be used and the possible values these can be assigned. For every tuple of values from the domain D (corresponding to an assignment for the variables in the constraint), an element from the c-semiring is associated. This element can be thought of as a weight or as a level of confidence in the assignment.

DEFINITION 3.2.4 (Constraint). Given a constraint system CS = $\langle \mathcal{S}, V, D\rangle$, a *constraint* c over CS is a pair $\langle con, def\rangle$, where

- *con* is a subset of V, called the *type* of the constraint,
- $def : [con \to D] \to S$ is a function from the set of all possible assignments (valuations) for the variables of c into the underlying set of the c-semiring; this is called the *satisfaction value function* of the constraint.

A *constraint problem* consists of a set of constraints defined over a constraint system and a set of variables of interest, i.e. a set of variables for which we want to find assignments that satisfy with a value greater than 0, all the given constraints.

DEFINITION 3.2.5 (Constraint problem). Given a constraint system CS $= \langle \mathcal{S}, V, D \rangle$, a *constraint problem* P over CS is a pair $\langle C, pv \rangle$, where C is a set of constraints over CS, and $pv \subseteq \cup\{con \mid \langle con, def \rangle \in C\}$ is the set of all variables in C.

The satisfaction values specified for the set of possible assignments for every constraint of a problem P are used to compute the satisfaction values for the possible assignments for the variables in pv by applying the operations of the c-semiring. The multiplicative operation $\times$ is used for defining a *combination* operation $\otimes$ on constraints, which combines satisfaction values of assignments for the variables of each constraint in order to obtain a satisfaction value of an assignment for all the variables of the problem P. The additive operation $\sum$ is used for defining a *projection* operation $\Downarrow$ on constraints, which computes the satisfaction value of the assignments (only) for the variables specified in the type of the problem P.

DEFINITION 3.2.6 (Combination operation). Given a constraint system CS $= \langle \mathcal{S}, V, D \rangle$ and two constraints $c_1 = \langle con_1, def_1 \rangle$ and $c_2 = \langle con_2, def_2 \rangle$ over CS, their *combination*, denoted $c_1 \otimes c_2$, is the constraint $c = \langle con, def \rangle$ with $con = con_1 \cup con_2$, and $def(\chi) = def_1(\chi\restriction_{con_1}) \times def_2(\chi\restriction_{con_2})$ for every valuation χ, where by $\chi\restriction_{con_i}$ we denote the projection of the tuple χ of assigned elements from D to the set of variables con_i of the constraint c_i.

$$\begin{tikzcd}
con_i \arrow[r, "\subseteq"] \arrow[rr, bend right, "\chi\restriction_{con_i}"'] & con \arrow[r, "\chi"] & D
\end{tikzcd}$$

Notice that the combination operation is commutative and associative. It can be extended straightforward to finite sets of constraints $C = \{c_1, \ldots, c_n\}$; we denote by $\bigotimes$ C the result of $c_1 \otimes c_2 \otimes \cdots \otimes c_n$.

DEFINITION 3.2.7 (Projection operation). Given a constraint system CS $= \langle \mathcal{S}, V, D \rangle$, a constraint $c = \langle con, def \rangle$ over CS, and a set of variables $W \subseteq V$, the *projection* of c over W, denoted $c \Downarrow_W$, is the constraint $\langle con', def' \rangle$ with $con' = W \cap con$, and $def'(\chi') = \sum(\{def(\chi) \mid \chi \colon con \longrightarrow S, \chi\restriction_{con'} = \chi'\})$ for every valuation χ.

The *solution* of a problem P is defined as the constraint induced on the variables in pv by all the constraints of P.

DEFINITION 3.2.8 (Solution). Given a soft-constraint problem $P = \langle C, pv \rangle$ over a constraint system CS, the *solution* of P is the constraint given by $(\bigotimes C) \Downarrow_{pv}$.

Note that since the problem P refers only to variables in the constraint system CS, its solution is a constraint over the same set of variables as P. The solution of a problem P gives, for every valuation χ of the variables in pv, a satisfaction value in S. The *best level of consistency* of P is the best satisfaction value (greatest value according to the order $\leq$) among those associated with the valuations.

DEFINITION 3.2.9 (Best level of consistency). Given a soft-constraint problem $P = \langle C, pv \rangle$ over a constraint system CS, we define best(P) as the (unique) value in S that corresponds to the constraint $(\bigotimes C) \Downarrow_{\emptyset}$ over the empty set of variables. If $0 <$ best(P), we say that P is *consistent*.

REMARK 3.2.10. In the VCSP framework, a constraint is defined as a function that assigns a cost to every possible valuation (assignment of values) for the variables. The costs are elements of underlying sets of the so-called *valuation structures*. A valuation structure is a tuple $\langle S, \geq +, 0, \infty \rangle$, where S is a totally ordered set (by $\geq$) with a minimum and a maximum element – 0 and ∞, and $+$ is a binary, commutative, and associative aggregation operation with unit 0, such that $a + c \geq b + c$ if $a \geq b$, for every $a, b, c \in S$. Although very similar to SCSP (modulo the algebraic structure adopted), the framework based on valued constraints is usually presented as a problem of cost minimization, rather than a problem of satisfaction maximization: the solution of an instance of a constraint problem is a valuation for the variables having a minimal cost. This does not match the intuition of the soft-constraint systems that we aim to explore, hence our preference for SCSP.

In what follows, we focus on the construction of a particular type of $\mathbb{RL}$-institution that is suitable for specifying soft-constraints satisfaction problems. Our choice of residuated lattices as truth structures for evaluating soft constraints is motivated by the fact that the addition of a residual operation to c-semirings and valuation structures has been shown in [BG06; Bov09] to provide a unifying framework for soft CSP: residuated lattices generalize both idempotent c-semirings and the so-called *fair valuation structures*, which are the structures usually employed with local consistency techniques [Bis+99]. We start with an example, and describe in detail how constraint specifications can be written over the Boolean institution $\mathcal{FOL}$ of first-order logic. This allows us to identify the properties and the additional structure that an institution $\mathcal{I}$ should have in order to deal with soft constraints, and to further define a many-valued institution CSP($\mathcal{I}$) of soft-constraint satisfaction problems based on $\mathcal{I}$.

THE FIRST-ORDER SOFT-CONSTRAINT $\mathbb{RL}$-INSTITUTION

To specify systems using constraints evaluated over residuated lattices, we consider first-order presentations that extend the presentation $\langle \Sigma_{\mathbb{L}}, E_{\mathbb{L}} \rangle$ of residuated lattices from Figure 2.1 – in other words, presentations $\langle \Sigma, E \rangle$ with $\Sigma_{\mathbb{L}} \subseteq \Sigma$ and $E \vDash E_{\mathbb{L}}$[3]. This means that, on the one hand, every $\langle \Sigma, E \rangle$-model has an underlying residuated lattice (its reduct as a $\Sigma_{\mathbb{L}}$-model) and that, on the other hand, we can make use of the symbols in $\Sigma_{\mathbb{L}}$ when writing the sentences of E. In this context, the only morphisms of presentations $\varphi \colon \langle \Sigma, E \rangle \longrightarrow \langle \Sigma', E' \rangle$ that we admit are those that do not change the symbols of $\Sigma_{\mathbb{L}}$.

```
spec Nat =
     sort   Nat
     ops    0 : Nat
            s : Nat ⟶ Nat
            _ + _ : Nat Nat ⟶ Nat    [comm assoc]
     pred   _ ≤ _ : Nat Nat

     ∀m, n : Nat
     • 0 + n = n
     • s(m) + n = s(m + n)
     • 0 ≤ m
     • s(m) ≤ s(n) if m ≤ n
```

FIGURE 3.1. The presentation Nat of natural numbers

EXAMPLE 3.2.11. Figure 3.2 depicts the specification of a customer's book-buying preferences written in the sublogic of $\mathcal{FOL}$ obtained by including in all presentations the basic specification ResiduatedLattices given in Figure 2.1. The specification BookData concerns a book trader that stores a number of books and offers three kinds of delivery: standard, express and online; for every book, two operations define the language in which a book is written and the number of days associated with each delivery mode. For that purpose, BookData extends the specification Nat in Figure 3.1 of natural numbers, which includes the usual successor and addition operations and a predicate $\leq$ corresponding to the ordering of natural numbers. Note

[3] Note the implicit translation of $E_{\mathbb{L}}$ from $\Sigma_{\mathbb{L}}$ to Σ.

that, in order to properly specify natural numbers in such an inductive fashion, one would need to add an initiality requirement to the specification. This falls outside of the expressive power of first-order presentations, but it can be obtained by using more complex structured specifications like those in [ST12]. It would be possible to extend our framework to structured specifications because the properties of presentations used in this thesis are also shared by structured specifications. However, for simplicity, we choose not to present that extension. In the specific case of natural numbers, for notational convenience, we abbreviate as n the term of the form $\mathsf{s}^n(0)$, i.e. n applications of the successor operation to 0.

```
logic FOL(ResiduatedLattices)[4]
spec BookData = Nat   then
        sorts  Book, Language, Delivery
        ops    en, fr, de, pt, ro, es : Language
               stnd, express, online : Delivery
               Schiele, ChagallMaVie, Munch : Book
               language : Book ⟶ Language

        • language(Schiele) = de
        • language(ChagallMaVie) = fr
        • language(Munch) = en

logic FOL(ResiduatedLattices)
spec Customer = BookData   then
        ops    deliveryTime : Book × Delivery ⟶ Nat
               languagePref : Language ⟶ L
               deliveryPref : Book × Delivery × Nat ⟶ L

        ∀ b : Book; n, n′ : Nat
        • languagePref(en) ≤ languagePref(de)
        • languagePref(de) ≤ languagePref(fr)
        • deliveryPref(b, express, n) ≤ deliveryPref(b, online, n′)
        • deliveryPref(b, stnd, n) ≤ deliveryPref(b, online, n′)
        • deliveryPref(b, express, n) ≤ deliveryPref(b, stnd, n′) if 3 ≤ n ∧ n′ ≤ 7
```

FIGURE 3.2. The presentations BookData and Customer

Customer extends the specification BookData by adding two new operation symbols, languagePref and deliveryPref, both of sort L introduced in the presentation of residuated lattices – which corresponds to their underlying set. Because every model of L is a residuated lattice, the two new oper-

[4] We denote by FOL(ResiduatedLattices) the sublogic of $\mathcal{FOL}$ obtained by including in all presentations the specification ResiduatedLattices from Figure 2.1, and not the $\mathbb{RL}$-institution $\mathcal{FOL}_{\mathbb{RL}}$.

ation symbols can be used to express preferences through axioms of the specification:

- German is preferred to English and French to German;
- regardless of the book and delivery time, online delivery is preferred to express and to standard;
- standard delivery is preferred to express when express delivery takes three days or more and standard takes seven days or less.

In order to include constraints in specifications, we need a new syntactic category through which we can declare *constraint variables*, and we need *constraint sentences* through which we can express preferences over those variables that we wish to be optimised. For example, in the case of Customer, we could specify the following constraint variables and sentences:

```
cvars book : Book; delivery : Delivery
• languagePref(language(book))    %(constraint)%
• deliveryPref(book, delivery, deliveryTime(book, delivery))    %(constraint)%
```

A constraint sentence (or *constraint*, for short) is a term of sort L. The specified constraints express the existence of preferences on the language in which the book is written, and the wish to optimise the method of delivery relatively to the expected delivery period. This optimisation is made relative to the axiomatization of the preferences in Customer: given a model of Customer and a valuation χ of the constraint variables (i.e. a choice of a book and of a delivery mode), every constraint is assigned a value (degree of satisfaction) in the residuated lattice; the degree of satisfaction of a constraint in a model can then be defined as the supremum of all the degrees of satisfaction obtained by varying χ, i.e. for all possible combinations of books and delivery modes, which in soft CSP is known as *the best level of consistency*.

The extension of first-order logic with constraint sentences is best accommodated in what are called *stratified institutions* [AD07], which provide an elegant way of capturing the valuations of constraint variables as states of models:

DEFINITION 3.2.12 (Stratified institution). A *stratified institution* $\mathcal{I}$ consists of:

- a category $\mathrm{Sig}^{\mathcal{I}}$, a functor $\mathrm{Sen}^{\mathcal{I}}$, and a functor $\mathrm{Mod}^{\mathcal{I}}$ as for an institution,
- a *stratification* $[\![\_]\!]^{\mathcal{I}}$, i.e. a collection of:
  - functors $[\![\_]\!]^{\mathcal{I}}_{\Sigma}\colon \mathrm{Mod}^{\mathcal{I}}(\Sigma) \to \mathrm{Set}$ for every signature Σ, giving for every Σ-model a set of *states*, and

  - surjective[5] natural transformations $[\![\_]\!]^{\mathcal{I}}_{\varphi}\colon [\![\_]\!]^{\mathcal{I}}_{\Sigma'} \Rightarrow \mathrm{Mod}^{\mathcal{I}}(\varphi)\,;[\![\_]\!]^{\mathcal{I}}_{\Sigma}$ for every signature morphism $\varphi\colon \Sigma \to \Sigma'$,

- for every signature Σ, a satisfaction relation between Σ-models and Σ-sentences that is parameterized by model states[6],

such that the satisfaction condition

$$\mathrm{Mod}^{\mathcal{I}}(\varphi)(M') \vDash_{\Sigma}^{[\![M']\!]^{\mathcal{I}}_{\varphi}(m')} \rho \qquad \text{if and only if} \qquad M' \vDash^{m'}_{\Sigma'} \mathrm{Sen}^{\mathcal{I}}(\varphi)(\rho)$$

holds for every signature morphism $\varphi\colon \Sigma \to \Sigma'$, Σ'-model M', model state $m' \in [\![M']\!]^{\mathcal{I}}_{\Sigma'}$ and Σ-sentence ρ.

THE STRATIFIED INSTITUTION OF FIRST-ORDER LOGIC. The stratified version of the Boolean institution $\mathcal{FOL}$ of first-order logic that we adopt, which we denote by $\underline{\mathcal{FOL}}$, has as signatures pairs $\langle \Sigma, V\rangle$ of a first-order-logic signature Σ and a set of sorted constraint variables V.

The $\langle \Sigma, V\rangle$-sentences are simply sentences over Σ with the constraint variables V regarded as new constants (nullary operation symbols).

The models of a signature $\langle \Sigma, V\rangle$ are the Σ-models from $\mathcal{FOL}$, while the states of a model M are the valuations $\chi\colon V \to M$, i.e. sorted functions from V to the many-sorted carrier set of M.

The satisfaction of a $\langle \Sigma, V\rangle$-sentence ρ by a $\langle \Sigma, V\rangle$-model M in a state χ is defined as the satisfaction of ρ in $\langle M, \chi\rangle$, i.e. in the extension of M for which the elements of V are interpreted according to χ.

REMARK 3.2.13. Notice that every presentation over the institution $\mathcal{FOL}$ of first-order logic can be seen as a presentation over $\underline{\mathcal{FOL}}$ by choosing an empty set of constraint variables, i.e. we can identify a first-order specification such as $\langle \Sigma_{\mathbb{L}}, E_{\mathbb{L}}\rangle$ with $\langle\langle \Sigma_{\mathbb{L}}, \emptyset\rangle, E_{\mathbb{L}}\rangle$.

We can now summarise in more rigorous terms the construction of the $\mathbb{RL}$-institution CSP($\underline{\mathcal{FOL}}$) of first-order soft-constraint satisfaction problems.

SIGNATURES. A signature is a pair $\langle \mathcal{L}, \Delta\rangle$ of a complete residuated lattice $\mathcal{L}$ and an extension $\Delta\colon \langle \Sigma_{\mathbb{L}}, E_{\mathbb{L}}\rangle \to \langle \Sigma, V, E\rangle$ of the presentation of residuated lattices. We include a residuated lattice in the signature in order to let specifiers decide the space of degrees of satisfaction they want to work with.

[5] By the surjectivity of the natural transformations we understand that for every morphism $\varphi\colon \Sigma \to \Sigma'$ and every Σ'-model M', the function $[\![M']\!]^{\mathcal{I}}_{\varphi}$ is surjective.

[6] To simplify the notation, we often omit the stratified institution in the super-script of the satisfaction relation, and we denote by $M \vDash^{m}_{\Sigma} \rho$ the satisfaction of ρ by M in a state m of M.

For CSP($\underline{\mathcal{FOL}}$), all extension morphisms Δ that we consider are inclusions. For simplicity, we denote $\langle \mathcal{L}, \Delta \colon \langle \Sigma_{\mathbb{L}}, E_{\mathbb{L}} \rangle \to \langle \Sigma, V, E \rangle \rangle$ by $\langle \mathcal{L}, \Sigma, V, E \rangle$. We continue to use the notation Δ only in model reducts, for brevity.

CONSTRAINT SENTENCES. A *constraint sentence* (or constraint, for short) for a signature $\langle \mathcal{L}, \Sigma, V, E \rangle$ is a $\langle \Sigma, V \rangle$-term of sort L.

MODELS. The models of $\langle \mathcal{L}, \Sigma, V, E \rangle$ are the models of $\langle \Sigma, V, E \rangle$ whose reducts along Δ are complete and admit a morphism into $\mathcal{L}$. It would be too restrictive to choose only those models of $\langle \Sigma, V, E \rangle$ whose reducts to $\Sigma_{\mathbb{L}}$ are $\mathcal{L}$ because we wish to support mappings between specifications that use different residuated lattices as their spaces of degrees of satisfaction. Formally, a model of $\langle \mathcal{L}, \Delta \colon \langle \Sigma_{\mathbb{L}}, E_{\mathbb{L}} \rangle \to \langle \Sigma, V, E \rangle \rangle$ is a pair $\langle M, f \rangle$ consisting of a model M of $\langle \Sigma, V, E \rangle$ together with a morphism $f \colon M {\upharpoonright}_{\Delta} \to \mathcal{L}$.

TRUTH SPACE. For every signature $\langle \mathcal{L}, \Delta \rangle$, the associated truth-space is simply its underlying lattice $\mathcal{L}$.

SATISFACTION RELATION. For every constraint signature $\langle \mathcal{L}, \Sigma, V, E \rangle$ and every model M, we define the value of a constraint c over M as the *best level of consistency*:

$$\big(\langle M, f \rangle \vDash_{\langle \mathcal{L}, \Sigma, V, E \rangle} c\big) = f\big(\textstyle\bigvee_{\chi \in [\![M]\!]_{\Sigma}} \mathrm{eval}_{\langle M, \chi \rangle}(c)\big),$$

where $\mathrm{eval}_{\langle M, \chi \rangle}(c)$ is the usual (inductively defined) interpretation of the first-order $\langle \Sigma, V \rangle$-term c in the model $\langle M, \chi \rangle$. Note that f translates the supremum to the residuated lattice $\mathcal{L}$ chosen by the specifier.

THE $\mathbb{RL}$-INSTITUTION OF SOFT CSP OVER $\mathcal{I}$

We now generalize the construction CSP($\underline{\mathcal{FOL}}$) to an arbitrary stratified institution $\mathcal{I} = \langle \mathbb{S}\mathrm{ig}^{\mathcal{I}}, \mathrm{Sen}^{\mathcal{I}}, \mathrm{Mod}^{\mathcal{I}}, [\![\_]\!]^{\mathcal{I}}, \vDash^{\mathcal{I}} \rangle$ that satisfies the following conditions:

C1. To make complete residuated lattices available to the specifier, we require the existence of an $\mathcal{I}$-presentation $\langle \Sigma_{\mathbb{L}}, E_{\mathbb{L}} \rangle$ such that $\mathbb{RL} \subseteq \mathrm{Mod}^{\mathcal{I}\text{-pres}}(\Sigma_{\mathbb{L}}, E_{\mathbb{L}})$. This does not restrict the applicability of the framework that we propose because most institutions suitable for the domains where soft constraints are useful provide the ability to specify ordered structures.

C2. In order to be able to express constraints, we require the existence of a functor $\mathrm{C} \colon \mathbb{S}\mathrm{ig}^{\mathcal{I}} \to \mathbb{S}\mathrm{et}$ that provides the set of constraints for every signature. In addition, we assume that for every object $\Delta \colon \langle \Sigma_{\mathbb{L}}, E_{\mathbb{L}} \rangle \to \langle \Sigma, E \rangle$ of the comma category $\langle \Sigma_{\mathbb{L}}, E_{\mathbb{L}} \rangle / \mathbb{P}\mathrm{res}^{\mathcal{I}}$ there exists a family of functions

$|M|_\Sigma \colon [\![M]\!]_\Sigma \to [\mathrm{C}(\Sigma) \to M{\upharpoonright}_\Delta]$, indexed by Σ-models M, which map every state of a model to a function that evaluates constraints, such that the equality

$$|M'|_{\Sigma'}(\chi')(C(\varphi)(c)) \;=\; |M'{\upharpoonright}_\varphi|_\Sigma([\![M']\!]_\varphi(\chi'))(c)$$

holds for any signature morphism $\varphi \colon \Sigma \to \Sigma'$, Σ'-model M', state χ' of M' and constraint $c \in \mathrm{C}(\Sigma)$.

On this basis, we define the $\mathbb{RL}$-institution $\mathrm{CSP}(\mathcal{I})$ as follows:

SIGNATURES. The category $\mathbb{S}\mathrm{ig}^{\mathrm{CSP}(\mathcal{I})}$ of constraint signatures is the product category of $\mathbb{RL}^{\mathrm{op}}$ and the comma category $\langle \Sigma_{\mathbb{L}}, E_{\mathbb{L}} \rangle / \mathbb{P}\mathrm{res}^{\mathcal{I}}$. This means that for any signature morphism $\langle \ell, \varphi \rangle \colon \langle \mathcal{L}, \Delta \rangle \to \langle \mathcal{L}', \Delta' \rangle$, ℓ is a morphism between the complete residuated lattices $\mathcal{L}'$ and $\mathcal{L}$, and φ is a morphism of presentations between the codomains of Δ and Δ' that makes the following diagram commutative:

$$\begin{tikzcd}
 & \langle \Sigma_{\mathbb{L}}, E_{\mathbb{L}} \rangle \arrow[dl, "\Delta"'] \arrow[dr, "\Delta'"] & \\
\langle \Sigma, E \rangle \arrow[rr, "\varphi"'] & & \langle \Sigma', E' \rangle
\end{tikzcd}$$

SENTENCES. $\mathrm{Sen}^{\mathrm{CSP}(\mathcal{I})}(\mathcal{L}, \Delta \colon \langle \Sigma_{\mathbb{L}}, E_{\mathbb{L}} \rangle \to \langle \Sigma, E \rangle) = \mathrm{C}(\Sigma)$.

MODELS. $\mathrm{Mod}^{\mathrm{CSP}(\mathcal{I})}(\mathcal{L}, \Delta) = \mathrm{MRL}_\Delta / \mathcal{L}$, where the functor MRL_Δ is the restriction and corestriction of $\mathrm{Mod}^{\mathcal{I}\text{-pres}}(\Delta)$ from $\mathrm{Mod}^{\mathcal{I}\text{-pres}}(\Delta)^{-1}(\mathbb{RL})$, the subcategory of $\langle \Sigma, E \rangle$-models whose subjacent residuated lattices are complete, into $\mathbb{RL}$.

TRUTH SPACE. The truth-space functor $\mathrm{RL}^{\mathrm{CSP}(\mathcal{I})}$ is the forgetful functor that maps every signature $\langle \mathcal{L}, \Delta \rangle$ to its underlying lattice $\mathcal{L}$.

SATISFACTION. Given an $\langle \mathcal{L}, \Delta \rangle$-model $\langle M, f \colon M{\upharpoonright}_\Delta \to \mathcal{L} \rangle$ and a sentence ρ for the same signature, the satisfaction of ρ by $\langle M, f \rangle$ is defined as:

$$\big(\langle M, f \rangle \vDash^{\mathrm{CSP}(\mathcal{I})}_{\langle \mathcal{L}, \Delta \rangle} \rho\big) = f\big(\textstyle\bigvee_{\chi \in [\![M]\!]_\Sigma} |M|_\Sigma(\chi)(\rho)\big).$$

There is an obvious way to translate $\langle \mathcal{L}, \Delta \rangle$-sentences along a signature morphism $\langle \ell, \varphi \rangle$: we ignore the lattice homomorphism and translate sentences along φ as in the base institution. Correspondingly, the reduct of a model $\langle M', f' \rangle$ of $\langle \mathcal{L}', \Delta' \rangle$ along $\langle \ell, \varphi \rangle$, is defined as $\langle M'{\upharpoonright}_\varphi, f' \,;\, \ell \rangle$. The well-definedness of model reducts is ensured by the commutativity of the following diagram.

$$\begin{tikzcd}
M'{\restriction}_{\Delta'} \arrow[r, "\mathrm{id}_{M'\restriction\Delta'}"] \arrow[d, "f'"'] & (M'{\restriction}_{\varphi}){\restriction}_{\Delta} \arrow[d, "f'\,;\,\ell"] \\
\mathcal{L}' \arrow[r, "\ell"'] & \mathcal{L}
\end{tikzcd}$$

PROPOSITION 3.2.14. *For any stratified institution $\mathcal{I}$ satisfying the conditions C1 and C2 above,* CSP($\mathcal{I}$) *is an* $\mathbb{RL}$*-institution.*

PROOF. The functoriality of the translation of sentences and of the reduction of models follows immediately from the definitions. Therefore, we only need to focus on the satisfaction condition.

Let us consider a signature morphism $\langle \ell, \varphi \rangle \colon \langle \mathcal{L}, \Delta \rangle \to \langle \mathcal{L}', \Delta' \rangle$, an $\langle \mathcal{L}', \Delta' \rangle$-model $\langle M', f' \colon M'{\restriction}_{\Delta'} \to \mathcal{L}' \rangle$, and an $\langle \mathcal{L}, \Delta \rangle$-sentence ρ. To check the satisfaction condition

$$\mathrm{RL}(\ell, \varphi)\big(\langle M', f' \rangle \vDash_{\langle \mathcal{L}', \Delta' \rangle} \langle \ell, \varphi \rangle(\rho)\big) = \big(\langle M', f' \rangle{\restriction}_{\langle \ell, \varphi \rangle} \vDash_{\langle \mathcal{L}, \Delta \rangle} \rho\big),$$

we first notice that

$$\begin{aligned}
\mathrm{RL}(\ell, \varphi)(\langle M', f' \rangle \vDash_{\langle \mathcal{L}', \Delta' \rangle} \langle \ell, \varphi \rangle(\rho)) &= \ell(\langle M', f' \rangle \vDash_{\langle \mathcal{L}', \Delta' \rangle} \varphi(\rho)) \\
&= \ell\big(f'\big(\bigvee_{\chi' \in [\![M']\!]_{\Sigma'}} |M'|_{\Sigma'}(\chi')(\varphi(\rho))\big)\big) \qquad (3.1)
\end{aligned}$$

and

$$\begin{aligned}
\big(\langle M', f' \rangle{\restriction}_{\langle \ell, \varphi \rangle} \vDash_{\langle \mathcal{L}, \Delta \rangle} \rho\big) &= \big(\langle M'{\restriction}_{\varphi}, f\,;\,\ell \rangle \vDash_{\langle \mathcal{L}, \Delta \rangle} \rho\big) \qquad (3.2) \\
&= (f'\,;\,\ell)\big(\bigvee_{\chi \in [\![M'{\restriction}_{\varphi}]\!]_{\Sigma}} |M'{\restriction}_{\varphi}|_{\Sigma}(\chi)(\rho)\big) \\
&= \ell\big(f'\big(\bigvee_{\chi \in [\![M'{\restriction}_{\varphi}]\!]_{\Sigma}} |M'{\restriction}_{\varphi}|_{\Sigma}(\chi)(\rho)\big)\big). \qquad (3.3)
\end{aligned}$$

That (3.1) and (3.3) are equal follows from the surjectivity of $[\![M']\!]_{\varphi}$ and the last condition assumed to be satisfied by the stratified institution $\mathcal{I}$: for every $\chi' \in [\![M']\!]_{\Sigma'}$, $|M'|_{\Sigma'}(\chi')(\varphi(\rho)) = |M'{\restriction}_{\varphi}|_{\Sigma}([\![M']\!]_{\varphi}(\chi'))(\rho)$. □

The construction above permits us to easily replace first-order logic with other logics suitable for specifying soft-constraint satisfaction problems. Two obvious candidates would be relational logic and equational logic, whose stratified institutions $\underline{\mathcal{REL}}$ and $\underline{\mathcal{EQL}}$ can be obtained as subinstitutions of

$\underline{\mathcal{FOL}}$ in the same way in which the ordinary institutions $\mathcal{REL}$ and $\mathcal{EQL}$ can be obtained from $\mathcal{FOL}$, as presented in Example 2.2.8. The most significant difference between the two resulting CSP institutions is the presentation of lattices. For relational logic it is based on the axiomatization of the relation $\leq$, whereas in the case of equational logic it is based on the operations $\wedge$ and $\vee$. Note that only $\mathcal{EQL}$ is expressive enough to specify residuated lattices. Nonetheless, because in the condition C1 above we only require that $\mathbb{RL}$ is included in the category of models of the specification $\langle \Sigma_{\mathbb{L}}, E_{\mathbb{L}} \rangle$, our framework accommodates easily both cases.

CSP($\mathcal{I}$) inherits a number of properties of $\mathcal{I}$. We list only a few that are used in the constructions presented in the next chapter.

PROPOSITION 3.2.15. *If* $\mathbb{S}\mathrm{ig}^{\mathcal{I}}$ *is finitely cocomplete, then so is* $\mathbb{S}\mathrm{ig}^{\mathrm{CSP}(\mathcal{I})}$.

PROOF. Consider the following diagram of a product of categories:

$$\begin{tikzcd}
\mathbb{C} & \mathbb{C} \times \mathbb{D} \arrow[l, "\pi_{\mathbb{C}}"'] \arrow[r, "\pi_{\mathbb{D}}"] & \mathbb{D}
\end{tikzcd}$$

We recall that, for any pair of finitely cocomplete categories $\mathbb{C}$ and $\mathbb{D}$, any colimit of a (finite) diagram $D \colon \mathbb{I} \to \mathbb{C} \times \mathbb{D}$ can be obtained simply by pairing the colimits of the diagrams $D \mathbin{;} \pi_{\mathbb{C}} \colon \mathbb{I} \to \mathbb{C}$ and $D \mathbin{;} \pi_{\mathbb{D}} \colon \mathbb{I} \to \mathbb{D}$.

We can apply this general result to the category of signatures of CSP($\mathcal{I}$), because the category of constraint signatures is the product category of $\mathbb{RL}^{\mathrm{op}}$ and the comma category $\langle \Sigma_{\mathbb{L}}, E_{\mathbb{L}} \rangle / \mathbb{P}\mathrm{res}^{\mathcal{I}}$. By Proposition 2.3.13, the category $\mathbb{RL}$ has finite limits, and hence $\mathbb{RL}^{\mathrm{op}}$ has finite colimits.

Therefore, all we need to know is that $\langle \Sigma_{\mathbb{L}}, E_{\mathbb{L}} \rangle / \mathbb{P}\mathrm{res}^{\mathcal{I}}$ is finitely cocomplete. To this end, we rely on two other well-known general results:

1. For any object $C \in |\mathbb{C}|$, the comma category $C/\mathbb{C}$ inherits the colimits of $\mathbb{C}$.
2. The category $\mathbb{P}\mathrm{res}^{\mathcal{I}}$ is cocomplete whenever $\mathbb{S}\mathrm{ig}^{\mathcal{I}}$ is cocomplete [GB92].

□

PROPOSITION 3.2.16. *If* $\mathcal{I}$ *has (weak) model amalgamation, then so does* CSP($\mathcal{I}$).

PROOF. We have to show that every pushout square of constraint signature morphisms

$$\begin{tikzcd}
\langle \mathcal{L}, \Delta \rangle \arrow[r, "\langle \ell_1, \varphi_1 \rangle"] \arrow[d, "\langle \ell_2, \varphi_2 \rangle"'] & \langle \mathcal{L}_1, \Delta_1 \rangle \arrow[d, "\langle \ell_1', \varphi_1' \rangle"] \\
\langle \mathcal{L}_2, \Delta_2 \rangle \arrow[r, "\langle \ell_2', \varphi_2' \rangle"'] & \langle \mathcal{L}', \Delta' \rangle
\end{tikzcd}$$

is a model-amalgamation square, meaning that, for every $\langle \mathcal{L}_1, \Delta_1 \rangle$-model $\langle M_1, f_1 \rangle$ and every $\langle \mathcal{L}_2, \Delta_2 \rangle$-model $\langle M_2, f_2 \rangle$ such that $\langle M_1, f_1 \rangle \upharpoonright_{\langle \ell_1, \varphi_1 \rangle} = \langle M_2, f_2 \rangle \upharpoonright_{\langle \ell_2, \varphi_2 \rangle}$, there exists an $\langle \mathcal{L}', \Delta' \rangle$-model $\langle M', f' \rangle$ such that $\langle M_1, f_1 \rangle = \langle M', f' \rangle \upharpoonright_{\langle \ell'_1, \varphi'_1 \rangle}$ and $\langle M_2, f_2 \rangle = \langle M', f' \rangle \upharpoonright_{\langle \ell'_2, \varphi'_2 \rangle}$.

Following the definition of morphisms of constraint signatures, we denote the codomain of Δ by $\langle \Sigma, E \rangle$. Similar notational conventions apply to the presentation morphisms Δ_1, Δ_2 and Δ'. We recall that the model-amalgamation property can be generalized in a straightforward way from signatures to the presentations over $\mathcal{I}$ (see, for example, [Dia14]). This means that for any $\langle \Sigma_1, E_1 \rangle$-model M_1 and $\langle \Sigma_2, E_2 \rangle$-model M_2 such that $M_1 \upharpoonright_{\varphi_1} = M_2 \upharpoonright_{\varphi_2}$, there exists a $\langle \Sigma', E' \rangle$-model M' such that $M' \upharpoonright_{\varphi'_1} = M_1$ and $M' \upharpoonright_{\varphi'_2} = M_2$. It thus suffices to show that there exists a morphism $f' \colon M' \upharpoonright_{\Delta'} \to \mathcal{L}'$ such that $f' \mathbin{;} \ell'_1 = f_1$ and $f' \mathbin{;} \ell'_2 = f_2$. Since, by the definition of morphisms of constraint signatures, the diagram below commutes,

$$\begin{tikzcd}
 & \langle \Sigma_{\mathbb{L}}, E_{\mathbb{L}} \rangle \arrow[dl, "\Delta_2"'] \arrow[d, "\Delta'"] \arrow[dr, "\Delta_1"] & \\
\langle \Sigma_2, E_2 \rangle \arrow[r, "\varphi'_2"'] & \langle \Sigma', E' \rangle & \langle \Sigma_1, E_1 \rangle \arrow[l, "\varphi'_1"]
\end{tikzcd}$$

it follows that $M' \upharpoonright_{\Delta'} = M_1 \upharpoonright_{\Delta_1} = M_2 \upharpoonright_{\Delta_2}$. Furthermore, because $\mathcal{L}'$ is the vertex of a pullback of morphisms of residuated lattices, it follows that there exists a (unique) morphism $f' \colon M' \upharpoonright_{\Delta'} \to \mathcal{L}'$ that makes the following diagram commute:

$$\begin{tikzcd}
 & & \mathcal{L}_2 \arrow[dr, "\ell_2"] & \\
M' \upharpoonright_{\Delta'} \arrow[urr, bend left, "f_2"] \arrow[drr, bend right, "f_1"'] \arrow[r, dashed, "f'"] & \mathcal{L}' \arrow[ur, "\ell'_2"] \arrow[dr, "\ell'_1"'] & & \mathcal{L} \\
 & & \mathcal{L}_1 \arrow[ur, "\ell_1"'] &
\end{tikzcd}$$

□

Before presenting a result on obtaining a factorization system for the signatures of CSP($\mathcal{I}$) from one for the signatures of $\mathcal{I}$, we first present a result on lifting factorization systems from signatures to presentations.

REMARK 3.2.17. Let $\mathcal{I}$ be an institution such that its category of signatures $\mathbb{S}\mathrm{ig}^{\mathcal{I}}$ admits a factorization system $\langle \mathbb{E}, \mathbb{M} \rangle$. We can obtain a factorization system $\langle \mathbb{E}^{\mathrm{pres}}, \mathbb{M}^{\mathrm{pres}} \rangle$ for the category $\mathbb{P}\mathrm{res}^{\mathcal{I}}$ of $\mathcal{I}$-presentations, called a *strong presentation factorization system*,[7] as follows:

- the arrows of $\mathbb{E}^{\mathrm{pres}}$ are the morphisms of presentations $e\colon \langle \Sigma, E\rangle \to \langle \Sigma', E'\rangle$ for which the underlying signature morphisms $e\colon \Sigma \to \Sigma'$, epimorphisms of $\mathbb{E}$, satisfy $E' = e(E)$, and
- the arrows of $\mathbb{M}^{\mathrm{pres}}$ are the morphisms of presentations $m\colon \langle \Sigma, E\rangle \to \langle \Sigma', E'\rangle$ with the underlying signature morphisms $m\colon \Sigma \to \Sigma'$ the monomorphisms in $\mathbb{M}$.

Then we can factor any morphism of presentations $\varphi\colon \langle \Sigma, E\rangle \to \langle \Sigma', E'\rangle$ as in the diagram below, where $e\,;m$ is a factorization of φ, regarded as a plain signature morphism from Σ to Σ'.

$$\begin{tikzcd}
\langle \Sigma, E\rangle \arrow[rr, "\varphi"] \arrow[dr, "e"'] & & \langle \Sigma', E'\rangle \\
& \langle \varphi(\Sigma), e(E)\rangle \arrow[ur, "m"'] &
\end{tikzcd}$$

PROPOSITION 3.2.18. *Given two factorization systems $\langle \mathbb{E}_{\mathbb{RL}}, \mathbb{M}_{\mathbb{RL}}\rangle$ for $\mathbb{RL}$ and $\langle \mathbb{E}, \mathbb{M}\rangle$ for $\mathbb{S}\mathrm{ig}^{\mathcal{J}}$, we obtain a factorization system for $\mathbb{S}\mathrm{ig}^{\mathrm{CSP}(\mathcal{J})}$ by taking the epimorphisms to be the pairs of arrows in $\mathbb{M}_{\mathbb{RL}}$ and in $\langle \Sigma_{\mathbb{L}}, E_{\mathbb{L}}\rangle / \mathbb{E}^{\mathrm{pres}}$, and the monomorphisms to be the pairs of arrows in $\mathbb{E}_{\mathbb{RL}}$ and in $\langle \Sigma_{\mathbb{L}}, E_{\mathbb{L}}\rangle / \mathbb{M}^{\mathrm{pres}}$.*

PROOF. In what follows we use the results presented in Fact 2.1.4 from Section 2.1. We apply the first result to the category $\mathbb{RL}$ of residuated lattices, and thus obtain a factorization system for $\mathbb{RL}^{\mathrm{op}}$ from the one presented in Proposition 2.3.14, and the second result to the comma category $\langle \Sigma_{\mathbb{L}}, E_{\mathbb{L}}\rangle / \mathbb{P}\mathrm{res}^{\mathcal{J}}$ and the factorization system $\langle \mathbb{E}^{\mathrm{pres}}, \mathbb{M}^{\mathrm{pres}}\rangle$ for $\mathbb{P}\mathrm{res}^{\mathcal{J}}$ obtained from $\langle \mathbb{E}, \mathbb{M}\rangle$ along the lines of Remark 3.2.17. We can finally apply the third result to the category of constraint signatures, which is defined as the product category of $\mathbb{RL}^{\mathrm{op}}$ and $\langle \Sigma_{\mathbb{L}}, E_{\mathbb{L}}\rangle / \mathbb{P}\mathrm{res}^{\mathcal{J}}$, and obtain the factorization system $\langle \mathbb{E}_{\mathrm{CSP}}, \mathbb{M}_{\mathrm{CSP}}\rangle$, with $\mathbb{E}_{\mathrm{CSP}} = \mathbb{M}_{\mathbb{RL}} \times \langle \Sigma_{\mathbb{L}}, E_{\mathbb{L}}\rangle / \mathbb{E}^{\mathrm{pres}}$ and $\mathbb{E}_{\mathrm{CSP}} = \mathbb{E}_{\mathbb{RL}} \times \langle \Sigma_{\mathbb{L}}, E_{\mathbb{L}}\rangle / \mathbb{M}^{\mathrm{pres}}$. □

3.2.2 CONSTRAINING FREE JAZZ

The following case-study on free-jazz improvisation shows how many-valued logics enriched with the soft-constraint satisfaction framework can be employed to describe specific domains of computational creativity. To this end, we must go beyond the software-engineering context to explore

[7] The terminology is reminiscent to that of the strong presentation inclusion systems of [DGS93].

another area of algebraic specification, which is less known but equally far reaching in the way it addresses computational creativity: *semiotic spaces* as formalized in [Gog99] (we discuss a particular case of semiotic spaces – conceptual spaces – in detail in Section 6.2).

The music-improvisation model we present here falls, according to the taxonomy of motivations for the formalization and automation of music compositional processes [PMW02], in the domain of computational modelling of music cognition and musical creativity – see [WPM09]. This domain includes studies having both cognitive motivations and musicological goals that are not focused on generating aesthetically appealing music or obtaining useful compositional tools, but are rather interested in the degree in which a model serves the comprehension of the cognitive processes within composition and improvisation. Our main aim is not to propose and evaluate hypotheses on stylistic properties of jazz compositions. However, at a secondary level, our study lies at the intersection of computational modelling of musical styles and design of compositional tools: the framework could be implemented and thus used to create new computationally creative music systems.

FREE JAZZ. Defining free jazz is not easy, and it is usually done *per exclusionem*, as it is easier to be understood as the sum of things it is not, rather than of the things it is. A positive characterisation of free jazz with which we agree and therefore adopt in the present thesis was proposed in [MC08]: free jazz is the form of jazz in which the performers are the only ones held accountable for the music that is being played, since (generally) no standard notations[8] are followed. The music results from a dynamic, complex game that changes its rules throughout the performance. The success of the game is determined by the identity that emerges from both coherence and conflict – the emergent "dynamical orderings" of the music "that are both surprising and comprehensible" [BG05].

As highlighted in [Bor05], free jazz is by no means random or lacking rules, even if the evolution of an improvising act is a priori unpredictable due to its transforming constraints and rules: the standards of quality are high, albeit different from the ones of traditional music. Free improvisation is a form of music that makes its own rules. Free-jazz improvising is not typically pursuing the classical rhythms, harmonies, or melodies; its valuable aspects are rather the pervading creativity, the discovery of new musical dimensions, the emergence of a collective purpose, or the unexpected synchronizations that interrupt divergence moments. Moreover, the performances are of

[8] Through standard notation we understand both the prescriptive (scores) and the descriptive (transcriptions) conventional notation.

undefined length, ending when the musicians are satisfied with the already played music. These features seem to challenge (or disregard at least) the existing means of analysing and evaluating conventionally notated music; new tools are hence required for studying the free-jazz phenomenon.

Our aim is to focus on the dynamics of performances by means of formal, algebraic methods for complex systems. We formalize free-jazz performances starting from the idea that an improvisation can be seen as a collection of *music phase spaces* that organize themselves and, through blending, emerge as the performed music. In our model, the music improvised up to a point plays the role of a specification with constraints to be fulfilled, while all the music fragments that could continue the performance are seen as specifications that offer their intrinsic characteristics to satisfy the 'needs' of the ongoing music act. The purpose of this kind of soft CSP formalization is to find those specifications of music phase spaces that optimize the satisfaction of the constraints of the performance.

FREE-JAZZ SEMIOTICS: MUSIC PHASE SPACES. We follow the lines of [BG05] and regard improvisation processes as self-organizational systems of musical *phase spaces*. We consider that the continuous flow of a free improvisation can be segmented into musical sections (regions of a phase space) that capture a distinct musical feature or that have a certain level of cohesion – a prominent qualitative character that may be related to the rhythm, tempo, timbre etc (see Figures 3.3 and 3.4). The passage between these sections plays the important role of a bifurcation in the evolution of the modelled improvisation, and is referred to as a *phase transition*.

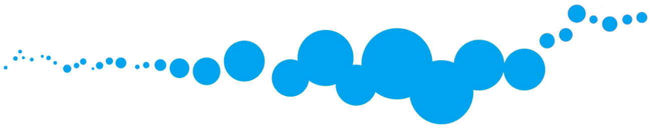


FIGURE 3.3. Representation of a music flow

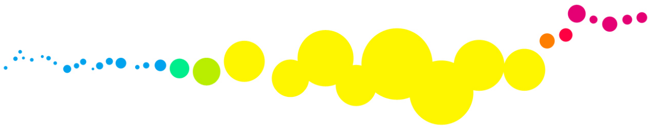


FIGURE 3.4. Segmentation of the music flow based on salient features

The phase space of a system is understood in [BG05] as a multi-dimensional map that facilitates the description of the dynamics of a given system.

The number of dimensions is given by the number of musical variables. The standard music notation captures, for example, a small number of dimensions: time, pitches, and marks regarding tempo or other details.

One might think that since free jazz does not commit to notations, and since it claims to be more flexible about tonalities and timbre, we would have to deal with phase spaces having large dimensions. What is proposed in [BG05] is actually a reduction of the unnecessary variables. We depart here, however, from the model proposed in [BG05], which is essentially a system of non-linear equations based on discrete time: phase spaces are just subspaces of high-dimensional state spaces, and the transitions between phase spaces are modelled through non-linear equations. We abstract over this representation of the phase spaces, and consider them simply as "musical idea spaces", or semiotic spaces of music. We loosen the algebraic formalization of semiotic spaces of Goguen [Gog99] by considering that a musical phase space could be described in principle through the use of algebraic specifications over a suitable logic. Thus, one should think of a music phase space as a collection of music fragments that share certain salient features and could be played at a certain moment in the evolution of the improvisation (see Figure 3.5).

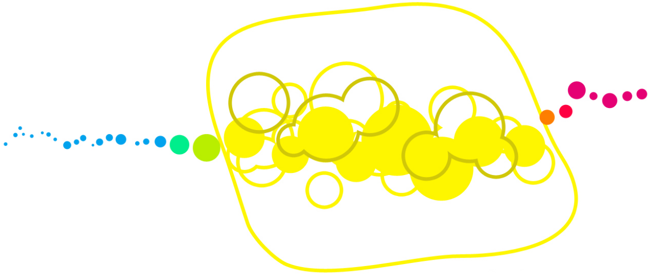


FIGURE 3.5. A segment of the music flow as part of a music phase space

The notion of concept blending as used by Goguen [Gog99] plays a key role in defining the composition of musical phase spaces, which in turn determines the outcome of the improvisation process. Similarly to the studies [Epp+15] and [KP+14] on the role of conceptual blending[9] in computational invention of cadences and chord progressions in jazz, we model the composition of musical phase spaces as categorical colimits of algebraic specifications. Examples of phase spaces, together with a formal definition of the notion, are discussed in the next subsection.

[9] Concept blending is presented in detail, together with conceptual or semiotic spaces, in Section 6.2; for this model of music improvisation, we restrict however the notion of blending to an ordinary colimit of presentations.

HUES OF SATISFACTION

We illustrate the formalization of free-jazz improvisations in the context of constraint specifications starting from the example presented in [BG05]. The authors analysed an excerpt entitled "Hues of Melanin" from the 1973 Sam Rivers Trio's concert at Yale University with Cecil McBee on bass and Barry Altschul on drums. They proposed a sectional interpretation of the performance and highlighted the transitions between the music phases in order to demonstrate the nonlinear dynamics of the improvisation. The segmentation is natural and determined by the frequent and clear variations of the music flow – rhythmic, timbral or chromatic. We focus on the first part of the examination in [BG05], namely sections A to H. Although the presentation of this example is self-explanatory, the reader is encouraged to consult the section "Hues of Melanin" of [BG05] for more details.

We consider that every musical section imposes some constraints regarding the tempo, texture, intensity, or technique details of the next musical phase to be played. However, we keep in mind that, as the improvisation builds, the constraints of a musical segment evolve and adapt to the already unfolded music: the same musical section or trigger of a transition could require different continuations if played at two different moments of the performance.

```
logic FOL(ResiduatedLattices)
spec FreeJazz =
        sorts Phase, Tempo, Texture, Instrument, Detail, Transition
        ops   slow, medium, fast : Tempo
              repetition, groove, complexity, fragmentation, rubato : Texture
              bass, drum, flute, sax, voice : Instrument
              trill, cadence, groove, drone, glissando, ascent, pedal : Detail
              N, T1, T2, T3, T4, T5, T6, T7 : Transition
              tempo : Phase ⟶ Tempo
              texture : Phase ⟶ Texture
              detail : Phase ⟶ Detail
              transition : Phase ⟶ Transition
```

FIGURE 3.6. The presentation FreeJazz

We zoom in on the two transitions triggered by a soprano saxophone trill on note D (sections C and G in Figure 3.7) and we regard the trill as a determining component in the evolution of the improvisation. For each of the sections C and G and their subsections, we record in Figure 3.7 the time of their start from the beginning of the performance, the transition type they initiate (and their actual trigger or salient detail in brackets), and their overall texture, when identified.

Name	Time	Transition Type	Overall Texture
C	5:29	T2 (soprano trill on note D)	FREE
C2	5:48	T7 (drum cadence)	
C3	6:43	T2 (soprano high note)	
C4	7:05	T2/T4/T6 (soprano, bass low A note)	
G	13:10	T2 (trill on note D), T5 (bass groove)	GROOVE
G2	14:04	metric sync	
G3	14:52	T6 (bass triggers descent)	FREE

Transition types: T2 *pseudo-cadential segue* – an implied cadence with sudden and unexpected continuation; T4 *feature overlap* – one feature of the antecedent section is sustained and becomes part of the consequent section; T5 *feature change* – a gradual change of one feature that redirects the flow (usually subtly); T6 *fragmentation* – a gradual breaking up, or fragmenting, of the general texture and/or rhythm; T7 *internal cadence* – a prepared cadence followed by a short silence then continuation with new material.

FIGURE 3.7. Sections and subsections of "Hues of Melanin" (excerpt from [BG05, Figure 1])

We model the musical phases of fragments C and G as specifications written over CSP($\underline{\mathcal{FOL}}$) that share a common sub-specification: the presentation FreeJazz in Figure 3.6, listing the instruments played by the musicians (through the constants bass, drum, flute, sax, voice), possible values for measuring the tempo (through the constants slow, medium, fast), texture descriptors (through the constants of sort Texture), techniques and ornamentations that constitute the salient details of musical segments (through the constants of sort Detail), as well as the types of transitions between the sections (through the constants T1–T6 and N of sort Transition, where the latter stands for the lack of a transition). Apart from these, a specification describing a musical phase also records the characteristics of the fragment and preferences on the musical section that will continue it. These are expressed as ordinary first-order sentences, such as the sentences of the presentation TriggeringPhase in Figure 3.8. In Section 3.3, we choose to make explicit the temporal distinction that separates them into properties of the current music fragment and properties of the next fragment.

This presentation corresponds to the musical space of music fragments

```
logic FOL(ResiduatedLattices)
spec TriggeringPhase = FreeJazz    then
     ops available : Phase × Instrument × Set [Instrument] ⟶ L
         tempoPref : Tempo ⟶ L
         texturePref : Texture ⟶ L
         instrumentPref : Instrument ⟶ L
         instrumentsPref : Set [Instrument] ⟶ L
         detailPref : Detail × Instrument ⟶ L
         transitionPref : Transition ⟶ L

     ∀ p : Phase; i : Instrument; is : Set [Instruments]
     • available(p, i, is) = 1 ⇔ (detail(p) = trill) ∧ (i = sax)
     • tempoPref(slow) ≤ tempoPref(medium)
     • tempoPref(fast) ≤ tempoPref(slow)
     • texturePref(complexity) ≤ texturePref(rubato)
     • texturePref(complexity) ≤ texturePref(groove)
     • instrumentPref(bass) = instrumentPref(drums) = instrumentPref(sax)
     • instrumentPref(flute) ≤ instrumentPref(sax)
     • instrumentsPref(is) = ∗(instrumentPref(is))
     • detailPref(trill, sax) = 0
     • transitionPref(N) = 0
```

FIGURE 3.8. The presentation TriggeringPhase

that can lead to the beginning of fragments C and G through the existence of a saxophone trill on the note D (notice the sentence $\text{available}(p, i, is) = 1 \Leftrightarrow (\text{detail}(p) = \text{trill}) \land (i = \text{sax})$ that identifies those music fragments containing a trill).

For the triggering passage we can distinguish a number of general constraints that do not depend on the context in which it is played, such as basic guidelines on the tempo, the texture, or the instruments to be played. Even though more specific preferences regarding the details of the section are left to be fixed at the actual moment of the musicking, there are some constraints regarding the need for a transition, and thus a continuation of the passage – the concert cannot end with the trill – and the repetition of the passage – it should not be continued with another saxophone trill. These restrictions are expressed as soft constraint sentences that extend the specification TriggeringPhase in Figure 3.8:

```
cvars phase : Phase;
      instrument : Instrument;
      instruments : Set [Instrument]

• tempoPref(tempo(phase))    %(constraint)%
• texturePref(texture(phase))   %(constraint)%
• instrumentsPref(instruments)   %(constraint)%
• transitionPref(transition(phase))   %(constraint)%
• detailPref(detail(phase), instrument)   %(constraint)%
```

These sentences restrict the entire musical phase space to several smaller phase spaces that satisfy the 'needs' of our musical component. In [BG05], the sections C and G that follow the saxophone trills on note D are considered to be alternatives for the development of the performance from our transitional point onwards: we can regard them as different solutions for a constraint problem. Although both sections satisfy the requirements imposed by playing the trill (the usual first-order sentences of the specification TriggeringPhase), the valuation of the constraint sentences above (which come with the already performed music) will discriminate between one choice and the other. We can think of sections C and G as epitomes of two growth directions or musical phase spaces: in the first phase, a medium-tempo short bass groove passage is soon abandoned for a rubato (the phase space fights the groove towards a modal area), while in the second section a groove similar to the one that was ended prematurely is explored (the phase space comprises passages of groove exploration and/or increased complexity).

The specifications PhaseSpaceC and PhaseSpaceG correspond to the two alternative phase spaces presented in our example. Each of these contains one of the two sections played during the actual improvisation.

```
logic FOL(ResiduatedLattices)
spec PhaseSpaceC = FreeJazz   then
     ops available : Phase × Instrument × Set [Instrument] ⟶ L
         p1, p2 : Phase

     ∀ p : Phase; i : Instrument; is : Set [Instrument]
     • available(p, i, is) = 1 ⇔ (p, i, is) belongs to the following table, [10]
                                  or 0 otherwise
```

phase (p)	tempo	texture	detail + instrument (i)	T	instruments (is)
p1	medium	rubato	cadence: drums	T7	drums, bass, sax
p2	slow	rubato	cadence: drums	T6	drums, bass

FIGURE 3.9. The specification PhaseSpaceC

The model we presented above captures only the static, structural aspects of "Hues of Melanin" – a counterpart of the representation from [BG05] of

[10] The table is only a convenient abbreviation for a set of sentences that specify, for example, that the music phase $p1$ has a medium tempo, rubato texture, its salient feature is a cadence on drums, its type of transition is T7, and the instruments involved are drums, bass and sax.

```
logic FOL(ResiduatedLattices)
spec PhaseSpaceG = FreeJazz    then
      ops available : Phase × Instrument × Set [Instrument] ⟶ L
            p1, p2, p3 : Phase
      ∀ p : Phase; i : Instrument; is : Set [Instrument]
      • available(p, i, is) = 1 ⇔ (p, i, is) belongs to the following table,
                                    or 0 otherwise
```

phase (p)	tempo	texture	detail + instrument (i)	T	instruments (is)
p1	medium	groove	drone: bass	T5	drums, bass, flute
p2	medium	groove	groove: bass	T5	drums, bass, sax
p3	fast	complex	trill: sax	T2	drums, bass, sax

FIGURE 3.10. The specification PhaseSpaceG

music segments as values for a collection of variables; the dynamics of the performance – a counterpart of the non-linear equations capturing the state transitions in [BG05] –, can be presented via the framework we describe in Chapter 4.

3.3 LOGICS FOR IMPROVISATION

First-order logic, although standard for formal specifications, may not be the most suitable choice for describing the features of a music fragment; we could specify music by employing other logics with a more convenient syntax, closer to the usual musical notation. Moreover, using CSP specifications to model music could be a rather excessive technical embellishment.

CHOOSING A LOGIC FOR SPECIFYING IMPROVISATION. In order to simplify the logical system used while continuing to ensure it is still expressive enough to serve the approach developed in the previous section on modelling music improvisations with algebraic specifications, we identify a set of informal requirements that a logic for improvisation must meet:

- It should permit the expression of constraints, as the aim of free jazz is "to play together with the greatest possible freedom – which, far from meaning without constraint, actually means to play together with sufficient skill and communication to be able to select proper constraints in the course of the piece"(musician Ann Farber, see [Bor05, Chapter "Reverence for Uncertainty"]).
- It should permit the partial satisfaction of these constraints, as improvisation requires flexibility and non-rigid answers.

- It should allow the change of the truth system, as players in a collaborative performance usually have different beliefs and value systems that they impose to the group alternately.

We argue that $\mathbb{RL}$-institutions in general satisfy the above criteria, and that they form an abstraction of CSP logics that is at times more suitable for modelling music-improvisation processes. Nonetheless, the many-valuedness of the constraints is certainly not the only feature that must be taken into account when choosing a logical system for specifying music. It is reasonable to assume that one must also consider:

- a balance between expressivity and complexity: while many-valued first-order logic is surely expressive enough for writing music scores, its complexity and decidability properties (or lack thereof) may guide us towards fragments of first-order logic – or to other logics that are equivalent to such fragments;
- support for dealing with sequences of triggers, events, or actions: an event could correspond to the fact that a certain music fragment has been played, while the action triggered by the event could be the play of a suitable continuation to that fragment. Obvious candidates for describing the succession in time of events and actions are temporal logic (see [Pnu77] or [Dia14] for a many-valued version) and dynamic logic [HKT01];
- the usability of notation: a musician could prefer a syntax more similar to the notation he uses for writing music. There is currently no logical system making use of specialised music notation that meets the requirements above. Therefore, non-standard notations used in composition notes and guide scores for improvised performances could be regarded as a starting point for defining the syntax of new logics.

Taking into account the above criteria, we propose a new logical system, formalized as an $\mathbb{RL}$-institution, based on Anthony Braxton's alternative notations for free-jazz 'composition'.

ANTHONY BRAXTON'S GRAPHIC NOTATIONS

Through extensive use of graphic and symbolic notations, Braxton's music positions itself at the fuzzy border between composition and improvisation [Loc08]. Neither completely notated, nor completely free, the scores can be seen as an incipient guideline for the improviser: the visual elements force the performer to assign personal interpretations to rather abstract forms that would otherwise make the scores unplayable by immutably following the more conventional notations. The improviser must hence intervene considerably in the composition, not in the usual form of jazz extended solos, but with "tiny pockets of improvisational space" [Loc08] that should fill the non-finished musical structure. This porosity, an inviting-to-improvisation characteristic of his compositions, is also apparent in the work on which we focus: "Composition 94 for Three Instrumentalists" [Bra88]. In section B of that piece, Braxton uses an *image grouping notation* consisting of three types of contours that are overlaid on top of standard pitches to create the so-called *liquid*, *shape*, and *rigid formations*.

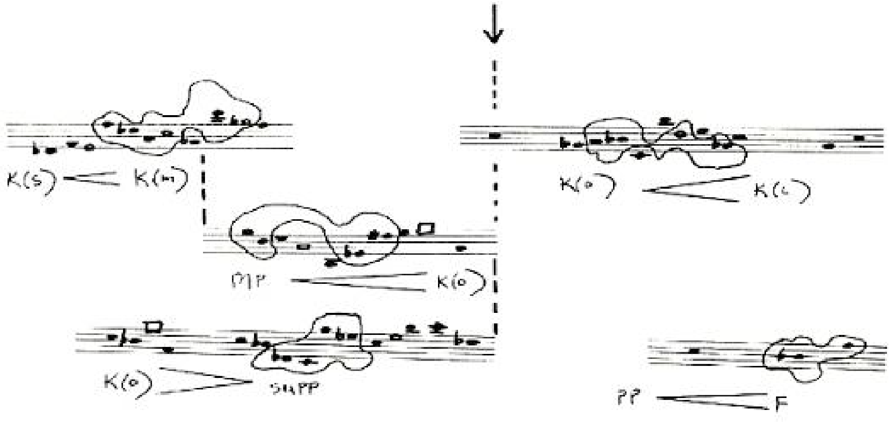


FIGURE 3.11. Liquid formations from the score of "Composition 94"

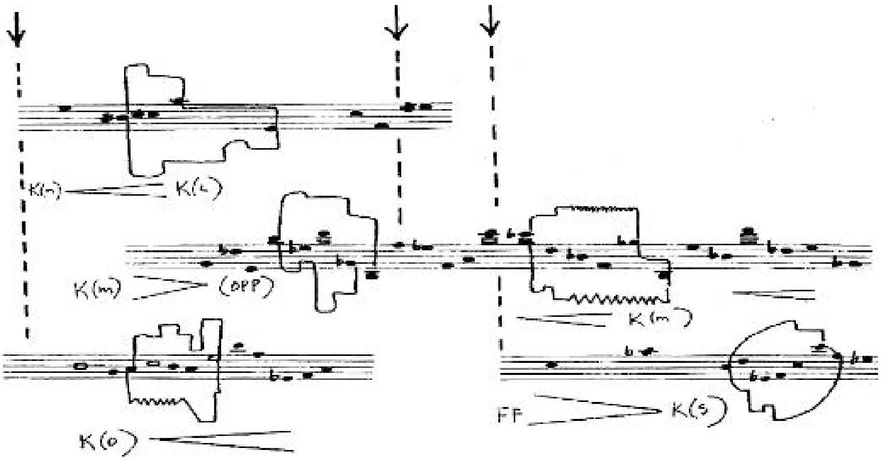


FIGURE 3.12. Shape formations from the score of "Composition 94"

The role of the formations is to indicate to the performers the outlines that they should follow when playing the notes inside them: these pitches

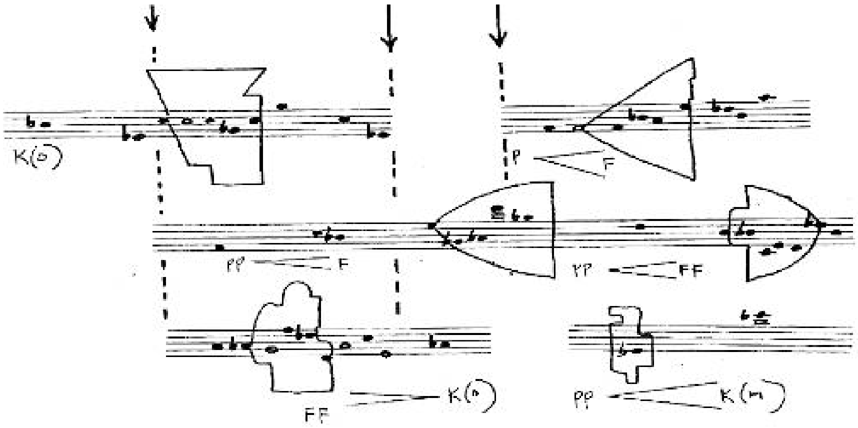


FIGURE 3.13. Rigid formations from the score of "Composition 94"

should not be played as they appear in the score, but transformed according to the distinctive interpretation of each improviser:

- the pitches inside liquid formations, figures resembling clouds, should be played as "clouded mass sound imprints" (Figure 3.11),
- the shape formations should suggest "harder edges" (Figure 3.12), while
- the rigid formations, closer to geometrical figures, should highlight their "composite state" (Figure 3.13).

In this study, we loosen the restrictions on the interpretation of formations, blurring the distinction between the three types of formations and, at the same time, making room for many other kinds of formations. We accept as valid the improvisations that replace the notes within the shapes with completely different pitches given that they allude to the original notes and evoke the contours. We reduce the problem of quantifying the improvisation's reminiscence of the original pitches to the problem of measuring the similarity between two music fragments. The interested reader is referred, for example, to [MS90] for further details on the comparison of musical fragments.

The observations above lead us to the formalization of Braxton's graphic notation as a many-valued logic.

THE $\mathbb{RL}$-INSTITUTION OF BRAXTON'S GRAPHIC NOTATION

SIGNATURES. To obtain a representation of the scores and, furthermore, to express properties of the music segments at certain positions in a score, the language of the logic of Braxton's graphic notation, which we denote by $\mathcal{BG}$, must comprise the universe of possible music fragments written in a standard notation such as classical music scores, a set of formations, and an appropriate truth structure that allows us to manipulate partially true statements. We hence define a $\mathcal{BG}$-*signature* Σ as a triple $\langle \mathcal{L}, \mathrm{MF}, \mathrm{FS} \rangle$, where $\mathcal{L}$ is a complete residuated lattice, MF is a set whose elements we call *music fragments*, and FS is a set of so-called *formation symbols*.

EXAMPLE 3.3.1 (Clap!). We accompany the definitions of the components of the institution with a very simple example, for which we choose to reduce to a minimum the details particular to music theory. We use an elementary notation for music made only by clapping. We assume that all claps are of equal intensity and have equal duration. This example will be later developed in Section 4.2, where we discuss Steve Reich's "Clapping Music" [Rei80]. For now, we consider a signature Σ, for which:

- the set MF consists of music fragments that can be written using pairs of symbols from the set {♪, 𝄾}. Each of these symbols denotes an action: ♪ for a clap, and 𝄾 for a rest. We use pairs to represent concurrent actions of two players – for example, two concurrent claps, or one clap and a rest, etc.;
- the set FS is empty – we use no formation symbols;
- the lattice is the Heyting algebra $\mathcal{HA}_\omega$ from Example 2.3.6 that has the underlying set of values $HA_\omega = \{0\} \cup \{1/n \mid n \in \omega\}$.

The *signature morphisms* $\varphi \colon \Sigma \to \Sigma'$ are defined componentwise:

- a morphism of residuated lattices $\varphi^\ell \colon \mathcal{L}' \to \mathcal{L}$,
- a function φ^{mf} between the sets of fragments MF and MF′,
- a function φ^{fs} from the set of formation symbols FS to FS′, and
- a natural number φ^d representing a delay between the moment of playing a score written over the first signature and the moment of playing a score written over the second one.

Usually, φ^{mf} and φ^{fs} are inclusions. Intuitively, this corresponds to transcribing music in a richer notation system. From the perspective of playing, the key component of φ is φ^d, which has an important role in glueing together music fragments.

EXAMPLE 3.3.2 (Clap!). We consider the addition of a formation symbol to our notation for clapping. We define the inclusion of the signature Σ to Σ', which has the same lattice $\mathcal{HA}_\omega$, same music fragments set MF, and a triangle formation symbol FS′ = {◺}.

SENTENCES. The *sentences* are built from atoms by applying the usual many-valued connectives. We admit three types of atomic sentences:

- $m@p$, with $m \in \mathrm{MF}$ and $p \in \mathbb{N}$, which should be read as "the fragment m occurs at position p",
- $\sim m@p$, with $m \in \mathrm{MF}$ and $p \in \mathbb{N}$, which should be read as "a fragment similar with m occurs at position p", and
- $s@p$, with $s \in \mathrm{FS}$ and $p \in \mathbb{N}$, which should be read as "the fragment at position p is in the shape of s".

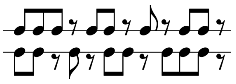
NOTATION. We denote by $s_p(m)$ the conjunction $s@p \wedge \sim m@p$.

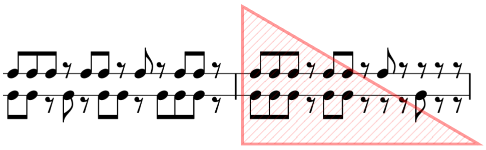
For any morphism of signatures $\varphi \colon \Sigma \to \Sigma'$, we can *translate* the atoms over Σ to atoms over Σ' as follows:

- $\varphi(m@p) = \varphi^{\mathrm{mf}}(m)@(p + \varphi^d)$,
- $\varphi(\sim m@p) = \sim\varphi^{\mathrm{mf}}(m)@(p + \varphi^d)$,
- $\varphi(s@p) = \varphi^{\mathrm{fs}}(s)@(p + \varphi^d)$,

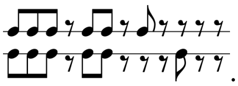
for every $m \in \mathrm{MF}$, $s \in \mathrm{FS}$, and $p \in \mathbb{N}$. This translation can be extended in a canonical way from atoms to arbitrary sentences.

EXAMPLE 3.3.3 (Clap!).

0  is an example of an atomic Σ'-sentence of the form $m@p$, with $p = 0$ and m a fragment with 12 beats, while the score



can be read as $(m@0) \wedge ($◺$@1) \wedge \sim(n@1)$, for n given by

.

MODELS. For any signature Σ, a *model* M consists of:

- a set $|M|$ whose elements are, intuitively, sounds produced by performers,
- interpretations $M_m \in |M|$ of the music fragments $m \in \mathrm{MF}$,
- a similarity method $M_\sim \colon |M| \times |M| \to \mathcal{L}$[11],

[11] One could impose additional restrictions on the similarity method, such as reflexivity, symmetry, or even a form of transitivity, where $a \sim c \leq (a \sim b) * (b \sim c)$.

- interpretations $M_s \colon |M| \to \mathcal{L}$ of the formation symbols $s \in \mathrm{FS}$, and
- an infinite sequence $M_{\mathrm{seq}} \in |M|^{\omega}$ of sounds to be played.

NOTATION. We usually denote the sound at position p in M_{seq} by $M_{\mathrm{seq}}[p]$, and the elements in M_{seq} from position p onward by $M_{\mathrm{seq}}[p..]$. The first element of the sequence is considered to be at position 0.

Model homomorphisms are defined as algebra homomorphisms: for two Σ-models M and N, a homomorphism $h \colon M \to N$ is a function $h \colon |M| \to |N|$ such that:

- $h(M_m) = N_m$,
- $M_s(x) \leq N_s(h(x))$,
- $M_{\sim}(x, y) \leq N_{\sim}(h(x), h(y))$, and
- $h(M_{\mathrm{seq}}) = N_{\mathrm{seq}}$ i.e the ordering of the elements in the sequence is kept,

for all $m \in \mathrm{MF}$, $s \in \mathrm{FS}$, and $x, y \in |M|$.

For every signature morphism $\varphi \colon \Sigma \to \Sigma'$, the *reduct* $M' \restriction_{\varphi}$ of a Σ'-model M' is defined as the Σ-model M such that:

- $|M| = |M'|$,
- $M_m = M'_{\varphi^{\mathrm{mf}}(m)}$ for $m \in \mathrm{MF}$,
- $M_{\sim} = M'_{\sim} \mathbin{;} \varphi^{\ell}$,
- $M_s = M'_{\varphi^{\mathrm{fs}}(s)} \mathbin{;} \varphi^{\ell}$ for $s \in \mathrm{FS}$, and
- $M_{\mathrm{seq}} = M'_{\mathrm{seq}}[\varphi^{d}..]$; in other words, M_{seq} is the suffix of M'_{seq} obtained by dropping from it the first φ^{d} elements.

EXAMPLE 3.3.4 (Clap!). Let us give an example of a Σ'-model M' for clapping. $|M'|$ should be thought of as the collection of possible sounds made by two musicians playing concurrently. We write every sound as a sequence of numbers representing how many claps are made on a beat: none, one, or two. Hence, $|M'|$ is the set $\{0, 1, 2\}^{*}$. Let us suppose that every music fragment $m \in \mathrm{MF}$ is interpreted as expected, with the number of claps indicated by the score. For example, the interpretations M'_m and M'_n of the music fragments m and n from Example 3.3.3 are 2211 1211 1221 and 2220 2201 0100, respectively.

The interpretation of the similarity measure is defined based on the number of different claps between the two sounds: $M'_{\sim}(a, b) = 1/\mathit{diff}$, where diff is the total number of different claps computed, element by element, for all beats of the two interpretations a and b. For example, the similarity of the interpretations M'_m and M'_n is $1/9$.

The interpretation of the shape ◺ is defined as $M'_{◺}(a) = 1/\mathit{diff}$, where diff is computed by comparing a with an 'ideal' triangle sequence. We use a moving window of size 3 to compute a's footprint of intensity: we calculate partial sums, each of three elements. Let us suppose that a is of length n. Then its footprint has length $n-2$. If $n-2$ is not divisible by 6 – the maximum value that we can obtain for the sum of a window (adding the maximum intensity of 2 claps 3 times), we extend the sequence a to the right with 0s. The length l of the footprint of the extended sequence is now divisible by 6. We compare the footprint with an ideal triangle-footprint sequence whose first $l/6$ elements have the value 6, the next $l/6$ elements the value 5, etc. We discard the possible extra 0s from the a's footprint and we search for the best match with a subsequence of the ideal footprint. The total number of differences between a's footprint and the ideal one gives us the value of diff. For example, $M'_{◺}(M'_m) = 1/14$ and $M'_{◺}(M'_n) = 1/5$, where the ideal footprint is 665544332211 and the footprints of m and n are (after discarding the 0s at the end) 54344443455 and 5444431211.

Let us define M'_{seq} as the infinite sequence of $(M'_m M'_n)^*$.

TRUTH SPACE. The *truth-space* functor is the forgetful functor that maps every signature to its underlying lattice.

SATISFACTION. For any signature Σ, the satisfaction of a sentence ρ by a Σ-model M is many-valued:

- $M \models m@p$ is defined as $\begin{cases} 1, & \text{if } M_{\text{seq}}[p] = M_m \\ 0, & \text{if } M_{\text{seq}}[p] \neq M_m \end{cases}$
- $M \models {\sim}m@p$ is given by the similarity of the interpretation of m and the sound at position p, i.e. $M_{\sim}(M_m, M_{\text{seq}}[p])$,
- $M \models s@p$ is given by the resemblance $M_s(M_{\text{seq}}[p])$ of the sound at position p with the shape s.

PROPOSITION 3.3.5. *$\mathcal{BG}$ is an $\mathbb{RL}$-institution.*

PROOF. The functoriality of the sentence translation and of the model reduction can be shown without difficulty, similarly to $\mathcal{FOL}_{\mathbb{RL}}$. For this reason, we we focus only on checking that the satisfaction condition

$$\varphi^{\ell}(M' \models_{\Sigma'} \varphi(\rho)) \;=\; (M' \upharpoonright_{\varphi} \models_{\Sigma} \rho)$$

holds for every signature morphism $\varphi\colon \Sigma \to \Sigma'$, Σ'-model M' and Σ-sentence ρ. And even in this case, it suffices to analyse the atomic sentences. Therefore, we consider three cases:

- for $\rho = m@p$

$$\begin{aligned}
\varphi^\ell(M' \models_{\Sigma'} \varphi(\rho)) &= \varphi^\ell(M' \models_{\Sigma'} \varphi^{\mathrm{mf}}(m)@(p + \varphi^d)) \\
&= \begin{cases} 1, & \text{if } M'_{\mathrm{seq}}[p + \varphi^d] = M'_{\varphi^{\mathrm{mf}}(m)} \\ 0, & \text{if } M'_{\mathrm{seq}}[p + \varphi^d] \neq M'_{\varphi^{\mathrm{mf}}(m)} \end{cases} \\
&= \begin{cases} 1, & \text{if } (M'\restriction_\varphi)_{\mathrm{seq}}[p] = (M'\restriction_\varphi)_m \\ 0, & \text{if } (M'\restriction_\varphi)_{\mathrm{seq}}[p] \neq (M'\restriction_\varphi)_m \end{cases} \\
&= M'\restriction_\varphi \models_\Sigma \rho
\end{aligned}$$

- for $\rho = {\sim}m@p$

$$\begin{aligned}
\varphi^\ell(M' \models_{\Sigma'} \varphi(\rho)) &= \varphi^\ell(M' \models_{\Sigma'} {\sim}\varphi^{\mathrm{mf}}(m)@(p + \varphi^d) \\
&= \varphi^\ell(M'_{\sim}(M'_{\varphi^{\mathrm{mf}}(m)}, M'_{\mathrm{seq}}[p + \varphi^d])) \\
&= (M'_{\sim} \mathbin{;} \varphi^\ell)(M_{\varphi^{\mathrm{mf}}(m)}, M'_{\mathrm{seq}}[p + \varphi^d]) \\
&= (M'\restriction_\varphi)_{\sim}((M'\restriction_\varphi)_m, (M'\restriction_\varphi)_{\mathrm{seq}}[p]) \\
&= M'\restriction_\varphi \models_\Sigma \rho
\end{aligned}$$

- for $\rho = s@p$

$$\begin{aligned}
\varphi^\ell(M' \models_{\Sigma'} \varphi(\rho)) &= \varphi^\ell(M' \models_{\Sigma'} \varphi^{\mathrm{fs}}(s)@(p + \varphi^d)) = \varphi^\ell(M'_{\varphi^{\mathrm{fs}}(s)}(M'_{\mathrm{seq}}[p + \varphi^d])) \\
&= (M'_{\varphi^{\mathrm{fs}}(s)} \mathbin{;} \varphi^\ell)(M'_{\mathrm{seq}}[p + \varphi^d]) = (M'\restriction_\varphi)_s((M'\restriction_\varphi)_{\mathrm{seq}}[p]) \\
&= M'\restriction_\varphi \models_\Sigma \rho
\end{aligned}$$

□

ENCODING $\mathcal{BG}$ INTO $\mathcal{FOL}_{\mathbb{RL}}$ WITH GENERALIZED SIGNATURE MORPHISMS

There exists a comorphism of $\mathbb{RL}$-institutions between $\mathcal{BG}$ and $\mathcal{FOL}^{\mathrm{gen}}_{\mathbb{RL}}$, the $\mathbb{RL}$-institution of first-order logic with generalized signature morphisms. We recall from [GP10] that a *generalized* $\mathcal{FOL}$*-morphism* between two signatures $\langle S, F, P\rangle$ and $\langle S', F', P'\rangle$ is a simple signature morphism between $\langle S, F, P\rangle$ and $\langle S', F' + T_{F'}, P'\rangle$, i.e. constants can be mapped to terms.

We define $\langle \Phi, \alpha, \beta, \lambda\rangle\colon \mathcal{BG} \to \mathcal{FOL}^{\mathrm{gen}}_{\mathbb{RL}}$, as follows:

SIGNATURES. The functor $\Phi\colon \mathbb{S}\mathrm{ig}^{\mathcal{BG}} \to \mathbb{S}\mathrm{ig}^{\mathcal{FOL}^{\mathrm{gen}}_{\mathbb{RL}}}$ maps

- every $\mathcal{BG}$-signature $\langle \mathcal{L}, \mathrm{MF}, \mathrm{FS}\rangle$ to the many-valued first-order signature $\langle \mathcal{L}, S, F, P\rangle$, where:
  - $S = \{\mathsf{Nat}\} \cup \{\mathsf{Mf}\}$, where Nat is the sort of natural numbers as in the presentation in Figure 3.1, and Mf is a new sort corresponding to music fragments;
  - $F = F_{\mathbb{N}} \cup \{m\colon \mathsf{Mf} \mid m \in \mathrm{MF}\} \cup \{@\colon \mathsf{Nat} \to \mathsf{Mf}\}$, where $F_{\mathbb{N}}$ are the operation symbols in the presentation of natural numbers;
  - $P = P_{\mathbb{N}} \cup \{s\colon \mathsf{Mf} \mid s \in \mathrm{FS}\} \cup \{\sim\colon \mathsf{Mf} \times \mathsf{Mf}\}$, where $P_{\mathbb{N}}$ are the relation symbols in the presentation of natural numbers.
- every signature morphism $\varphi\colon \langle \mathcal{L}, \mathrm{MF}, \mathrm{FS}\rangle \to \langle \mathcal{L}', \mathrm{MF}', \mathrm{FS}'\rangle$ to a many-valued first-order generalized signature morphism $\langle \ell, \phi\rangle\colon \langle \mathcal{L}, S, F, P\rangle \to \langle \mathcal{L}', S, F' + T_{F'}, P'\rangle$ between $\Phi(\mathcal{L}, \mathrm{MF}, \mathrm{FS}) = \langle \mathcal{L}, S, F, P\rangle$ and $\Phi(\mathcal{L}', \mathrm{MF}', \mathrm{FS}') = \langle \mathcal{L}', S, F', P'\rangle$ with:
  - $\ell = \varphi^{\ell}$;
  - $\phi^{\mathrm{st}} = \mathrm{id}_S$;
  - $\phi^{\mathrm{op}}(m) = \varphi^{\mathrm{mf}}(m)$ for all $m \in \mathrm{MF}$
$\phi^{\mathrm{op}}(@) = @$
$\phi^{\mathrm{op}}(0) = \varphi^{d}$ for the constant $0\colon \mathsf{Nat}$
$\phi^{\mathrm{op}}(f) = f$ for any other operation symbol f in $F_{\mathbb{N}}$;
  - $\phi^{\mathrm{rel}}(s) = \varphi^{\mathrm{fs}}(s)$ for all $s \in \mathrm{FS}$
$\phi^{\mathrm{rel}}(\sim) = \sim$
$\phi^{\mathrm{rel}}(\pi) = \pi$ for any other relation symbol π in $P_{\mathbb{N}}$.

SENTENCES. For every signature Σ, $\alpha_{\Sigma}\colon \mathrm{Sen}^{\mathcal{BG}}(\Sigma) \to \mathrm{Sen}^{\mathcal{FOL}^{\mathrm{gen}}_{\mathbb{RL}}}(\Phi(\Sigma))$ is the function defined, as usual, in an inductive manner, that maps every atom

- $m@p$ to the equational atom $@(p) = m$,
- $\sim m@p$ to the relational atom $m \sim @(p)$,
- $s@p$ to the relational atom $s(@(p))$,

for $p \in \mathbb{N}$, $m \in \mathrm{MF}$, and $s \in \mathrm{FS}$.

MODELS. For every signature Σ, $\beta_{\Sigma}\colon \mathrm{Mod}^{\mathcal{FOL}^{\mathrm{gen}}_{\mathbb{RL}}}(\Phi(\Sigma)) \to \mathrm{Mod}^{\mathcal{BG}}(\Sigma)$ is the functor that maps

- every first-order structure M' for $\Phi(\Sigma)$ to the Σ-model M with:
  - $|M| = M'_{\mathsf{Mf}}$,
  - $M_m = M'_m$ for all $m \in \mathrm{MF}$,

- $M_\sim = M'_\sim$,
- $M_s = M'_s$ for $s \in \mathrm{FS}$,
- $M_{\mathrm{seq}}[p] = M'_{@(p)}$ for $p \in \mathbb{N}$[12].

- every morphism of first-order structures $h' \colon M' \to N'$ in $\mathrm{Mod}^{\mathcal{FOL}^{\mathrm{gen}}_{\mathbb{RL}}}(\Phi(\Sigma))$ to a $\mathcal{BG}$-homomorphism $h \colon \beta_\Sigma(M') \to \beta_\Sigma(N')$ given by $h'_{\mathsf{Mf}} \colon M'_{\mathsf{Mf}} \to N'_{\mathsf{Mf}}$, the component of the homomorphism $h' \colon M' \to N'$ corresponding to the sort Mf. It is trivial to check that the additional conditions imposed on h are satisfied. For instance, for every natural number n, we have $h(\beta_\Sigma(M')_{\mathrm{seq}}[p]) = h'_{\mathsf{Mf}}(M'_@(p)) = N'_@(p) = \beta_\Sigma(N')_{\mathrm{seq}}[p]$.

Lattices. For every signature Σ, $\lambda_\Sigma \colon \mathrm{RL}^{\mathcal{FOL}^{\mathrm{gen}}_{\mathbb{RL}}}(\Phi(\Sigma)) \to \mathrm{RL}^{\mathcal{BG}}(\Sigma)$ is the identity functor.

Satisfaction condition. To check that the satisfaction condition holds, it suffices once again to consider only the atomic sentences. Let Σ be a $\mathcal{BG}$-signature, M a first-order structure for $\Phi(\Sigma)$, and ρ a sentence for Σ. We distinguish three cases:

- for $\rho = m@p$

$$\begin{aligned}
(M' \models^{\mathcal{FOL}\mathbb{RL}}_{\Phi(\Sigma)} \alpha_\Sigma(m@p)) &= (M' \models^{\mathcal{FOL}\mathbb{RL}}_{\Phi(\Sigma)} (@(p) = m)) \\
&= (M'_{@(p)} = M'_m) \\
&= \begin{cases} 1, & \text{if } M'_{@(p)} = M'_m \\ 0, & \text{if } M'_{@(p)} \neq M'_m \end{cases} \\
&= \begin{cases} 1, & \text{if } \beta_\Sigma(M')_{\mathrm{seq}}[p] = \beta_\Sigma(M')_m \\ 0, & \text{if } \beta_\Sigma(M')_{\mathrm{seq}}[p] \neq \beta_\Sigma(M')_m \end{cases} \\
&= \beta_\Sigma(M') \models^{\mathcal{BG}}_\Sigma m@p
\end{aligned}$$

- for $\rho = {\sim} m@p$

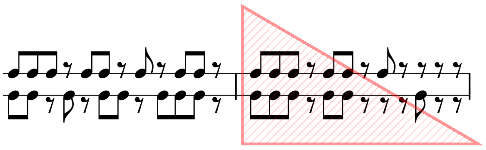
$$\begin{aligned}
(M' \models^{\mathcal{FOL}\mathbb{RL}}_{\Phi(\Sigma)} \alpha_\Sigma({\sim} m@p)) &= (M' \models^{\mathcal{FOL}\mathbb{RL}}_{\Phi(\Sigma)} (m \sim @(p))) \\
&= M'_\sim(M'_m, M'_{@(p)}) \\
&= \beta_\Sigma(M')_\sim(\beta_\Sigma(M')_m, \beta_\Sigma(M')_{\mathrm{seq}}[p]) \\
&= \beta_\Sigma(M') \models^{\mathcal{BG}}_\Sigma {\sim} m@p
\end{aligned}$$

[12] Note that, in the right-hand-side term $@(p)$, p is actually the term associated to the natural number p; it consists of p repeated applications of the successor operation to 0.

- for $\rho = s@p$

$$\begin{aligned}
(M' \models^{\mathcal{FOL}_{\mathbb{RL}}}_{\Phi(\Sigma)} \alpha_\Sigma(s@p)) &= (M' \models^{\mathcal{FOL}_{\mathbb{RL}}}_{\Phi(\Sigma)} s(@(p))) \\
&= M'_s(M'_{@(p)}) \\
&= \beta_\Sigma(M')_s(\beta_\Sigma(M')_{\mathrm{seq}}[p]) \\
&= \beta_\Sigma(M') \models^{\mathcal{BG}}_\Sigma s@p
\end{aligned}$$

EXAMPLE 3.3.6. Let us show how the sentence



– read as $(m@0) \wedge (◺@1) \wedge \sim(n@1)$ – from Example 3.3.3 can be encoded into a sentence over the $\mathbb{RL}$-institution $\mathcal{FOL}_{\mathbb{RL}}$.

First, we need to deal with the translation of its signature. The original $\mathcal{BG}$-sentence is written over the signature $\Sigma' = \langle \mathcal{HA}_\omega, \mathrm{MF}, \mathrm{FS}' = \{◺\}\rangle$ from Example 3.3.2. According to the definition of the institution comorphism above, Σ' is mapped to the first-order signature $\langle \mathcal{HA}_\omega, S, F, P\rangle$, where:

$S = \{\mathsf{Nat}, \mathsf{Mf}\}$,
$F = F_{\mathbb{N}} \cup \{m\colon \mathsf{Mf} \mid m \in \mathrm{MF}\} \cup \{@\colon \mathsf{Nat} \to \mathsf{Mf}\}$, and
$P = P_{\mathbb{N}} \cup \{\mathsf{triang}\colon \mathsf{Mf}$[13]$, \sim\colon \mathsf{Mf} \times \mathsf{Mf}\}$.

Then each of the three atoms of the $\mathcal{BG}$-sentence is translated as follows:

$$m@0 \;\mapsto\; @(0) = m \qquad ◺@1 \;\mapsto\; @(1) = \mathsf{triang} \qquad \sim(n@1) \;\mapsto\; n \sim @(1)$$

leading to the first-order sentence $(@(0) = m) \wedge (@(1) = \mathsf{triang}) \wedge (n \sim @(1))$.

SPECIFICATIONS OVER $\mathcal{BG}$

The fact that a signature does not determine the interpretation of the music fragments, the similarity measure, or the interpretation of the formation symbols, makes the logic $\mathcal{BG}$ too general for suitably specifying music: we would like to be able to control, for example, which similarity measure to use in comparing music fragments. For that purpose, and to continue to

[13] According to the definition of the comorphism, we should have kept the symbol ◺ instead of using the predicate triang. Yet we chose to change the symbol to show readers who are more familiar with the standard notation of first-order how one could capture music scores using regular first-order sentences.

keep the specification formalism as simple as possible, we introduce the logic $\mathcal{BGP}$ of that has as signatures presentations over $\mathcal{BG}$.

PROPOSITION 3.3.7. *$\mathcal{BGP}$ has the following properties:*

1. *$\mathcal{S}\mathrm{ig}^{\mathcal{BGP}}$ has pushouts.*
2. *$\mathcal{BGP}$ has model amalgamation.*
3. *There exists a factorization system for $\mathcal{S}\mathrm{ig}^{\mathcal{BGP}}$.*

PROOF.

1. Pushouts in $\mathcal{S}\mathrm{ig}^{\mathcal{BG}}$ are obtained componentwise, building on pushout constructions from $\mathbb{RL}^{\mathrm{op}}$ and $\mathcal{S}\mathrm{et}$. Moreover, because the category $\mathbb{P}\mathrm{res}^{\mathcal{I}}$ of presentations over an arbitrary $\mathbb{RL}$-institution $\mathcal{I}$ inherits the pushouts of $\mathcal{S}\mathrm{ig}^{\mathcal{I}}$, $\mathcal{S}\mathrm{ig}^{\mathcal{BGP}}$ has pushouts.
2. To show that $\mathcal{BGP}$ has model amalgamation, it suffices to verify that $\mathcal{BG}$ has model amalgamation, since the model-amalgamation property can be generalized in a straightforward way from signatures to presentations over $\mathcal{I}$, for an arbitrary $\mathbb{RL}$-institution $\mathcal{I}$, similarly to the Boolean version of institutions. For that purpose, consider the pushout square given by the $\mathcal{BG}$-signature morphisms φ, θ, σ_1 and σ_2 in the diagram below.

$$\begin{tikzcd}
\Sigma \arrow[r, "\varphi"] \arrow[d, "\theta"'] & \Sigma_1 \arrow[d, "\sigma_1"] \arrow[rd, bend left, "f"] & \\
\Sigma_2 \arrow[r, "\sigma_2"'] \arrow[rr, bend right, "g"'] & \Sigma' \arrow[r, "h"] & \Sigma''
\end{tikzcd}$$

Suppose that M_1 is a Σ_1-model and M_2 a Σ_2-model such that $\mathrm{Mod}(\varphi)(M_1) = \mathrm{Mod}(\theta)(M_2)$. Then we can define a Σ'-model M' (the amalgamation of M_1 and M_2) as follows: For the underlying set of sounds, $|M'| = |M_1| = |M_2|$. Now consider a music fragment $m' \in MF'$; since σ_1 and σ_2 form a pushout of θ and φ, there exists $i \in \{1, 2\}$ and $m_i \in \mathrm{MF}_i$ such that $\sigma_i(m_i) = m'$. Then we can define $M'_{m'}$ as $(M_i)_{m_i}$. Note that this does not depend on the choice of i or m_i, because on the one hand, σ_1, σ_2 form a pushout, and on the other hand, M_1 and M_2 have the same reduct to Σ. In the same (unique) manner, we can define the interpretations of shapes, similarities and of the sound sequence.
3. We define the factorization system for $\mathcal{S}\mathrm{ig}^{\mathcal{BG}}$ componentwise. Any signature morphism $\varphi \colon \Sigma \to \Sigma'$, given by components φ^{ℓ}, φ^{mf}, φ^{fs} and φ^{d}, can be

factored as $\varphi = e\,;m$

$$\begin{tikzcd}
\Sigma \arrow[rr, "\varphi"] \arrow[dr, "e"'] & & \Sigma' \\
& \varphi(\Sigma) \arrow[ur, "m"'] &
\end{tikzcd}$$

with $e = \langle m^{\ell}, e^{\mathrm{mf}}, e^{\mathrm{fs}}, e^{d}\rangle$ and $m = \langle e^{\ell}, m^{\mathrm{mf}}, m^{\mathrm{fs}}, m^{d}\rangle$, where $e^{d} = 0$, $m^{d} = \varphi^{d}$, and $\langle e^{\mathrm{mf}}, m^{\mathrm{mf}}\rangle$ and $\langle e^{\mathrm{fs}}, m^{\mathrm{fs}}\rangle$ belong to the standard factorization system for $\mathbb{S}$et, and $\langle e^{\ell}, m^{\ell}\rangle$ belongs to the factorization system $\langle \mathbb{E}_{\mathbb{RL}}, \mathbb{M}_{\mathbb{RL}}\rangle$ of $\mathbb{RL}$ from Proposition 2.3.14. We then lift the factorization-system thus obtained to presentations, as in Proposition 3.2.17.

□

4

REASONING FOR CREATIVE PROCESSES

In this chapter, we explore the algebraic and logical foundations of service-oriented computing and discuss its ramifications as a general paradigm for complex system dynamics that is particularly suited for computational creative processes. The first section is devoted to the definition of the concepts of service application and service module – which are central to service-oriented computing – over a fixed many-valued institution, and to the formalization of the mechanisms of service discovery, selection and binding. We then investigate the connection between service-oriented systems and computational creativity opened by our framework, while adopting the idea that music improvisations can be seen as instances of complex systems: we show how the concepts of service application and service module can be applied to the music phases spaces defined in Section 3.2.2, and how the dynamic processes of service-oriented computing can describe ad hoc music composition. Finally, we address new situations that arise in the context of heterogeneous and evolving service providers and requesters. Particular cases of interest are those in which different service components are based on different truth structures, or those for which preferences change over time. This brings clarity with respect to the uncertainty associated with predicting the evolutionary behaviour of service-oriented systems.

CREATIVE PROCESSES AS COMPLEX SYSTEMS. For modelling improvisation processes, we draw inspiration from [BG05] and [Bor05], where free-jazz performances are seen as non-linear dynamical systems of equations. Therein, the 'sink or swim' behaviour of a complex system is expressed as 'sync or swarm' for improvisation; in a nutshell, this means that we either select the states with the best fitness, or we decide to let a process end. The swarm formalization proposed by Borgo and Goguen aims to echo the emergence of a collective direction of the music performance as a whole, despite the seemingly divergence of its individual components.

The paper [RG94] is a proof of the long-standing interest in modelling and simulating the musical creativity of improvisations. The authors of the study make similar claims to our assumptions on knowledge and reasoning in improvisation performances: musical actions depend on contexts that evolve over time, and musicians integrate rules and constraints into their actions dynamically. Moreover, they set similar goals to those of this thesis with respect to simulating creativity: obtaining a suitable trade-off between the 'flexibility and randomness' and the 'control and clear semantics' in modelling creativity in terms of classical problem solving.

Even if music improvisation is almost never concerned with meeting objectives in an optimal way, there are a number of computational approaches in the literature that exploit optimization techniques to model a step in the evolution of the system; for example, improvisations are seen in [BY04] as swarm optimization processes. Our approach is somewhat aligned with the *Live Algorithms for Music* manifesto of [BY04], although we choose to depart from the randomness inherent to swarm optimization: the essential difference is that, in the modular view of these complex systems, we replace the *Swarmer* component with a service-oriented framework based on earlier work on algebraic formalizations of services such as [FLB06; FLB11].

4.1 SERVICE-ORIENTED COMPUTING

Service-oriented computing is a paradigm that supports the development of complex software applications based on dynamic reconfigurations of networks of systems. Reconfigurations arise from interactions between software entities and are governed by a need-fulfilment mechanism: during their execution, software applications may request to bind to external entities in order to fulfil the need for a service or resource. The non-mereological composition of requester applications and supplier services is mediated by *interfaces* through which system components require and guarantee the satisfaction of some quality-of-service conditions: a client first *discovers* the services that guarantee, through their interfaces, the satisfaction of its needs, and then *selects* and *binds* to a provider via a negotiation of *service level agreements*. The interfaces express properties that are independent of the actual implementation of the services, and are usually defined using algebraic specifications of abstract data types [FLB06; FLB11] or temporal-logic specifications [FL13b]. Besides the typical functional properties of the input-output behaviour of services described by the interfaces, one could also specify constraints that express preferences meant to be used for the selection of a best provider in terms of the maximization of their satisfaction degrees. Soft-constraint systems have been successfully employed for capturing non-functional requirements in service-oriented architectures [Wir+06; Wir+07;

HMW09], including the negotiation of service-level agreements [BS09].

The aim of this chapter is to formalize the three elementary processes of service discovery, selection and binding, and to reason about them using logical tools. Whereas two of these processes, namely discovery and binding, have been studied before over various logical frameworks, as in [FLB11; FL13b; ŢF15b], the selection has yet to receive a full logical treatment. This is mainly due to the fact that such a process would involve ranking all matching service providers (for a given request, at a given time, as in [FL13a]), and then selecting one that, by means of logical inference, offers the best possible match. The classical Boolean formalisms are not suitable for this task, as they can only provide yes/no answers; that is, we can determine if a service component meets the requirements of the application, but we cannot quantify the quality of service. For this reason, we choose to develop service-oriented processes over many-valued logical frameworks. An immediate solution would be to use the CSP($\mathcal{I}$) institutions defined in Section 3.2.1: one could describe service components by means of constraint specifications, and define the requirements of applications and the properties guaranteed by service providers as constraint sentences. However, we choose a more general approach based on arbitrary $\mathbb{RL}$-institutions, which allows us to explore the relationship between service-oriented systems and creative processes.

One of the main features of service-oriented computing that we address is that the actual architectural configuration of service-oriented systems (i.e. their components and the connectors through which components exchange information) cannot be anticipated at design time, because binding to new services could trigger subsequent processes of discovery and binding. Moreover, the dynamicity of the reconfigurations is endogenous, intrinsic to the systems, as their evolution is not driven by external factors such as the change of the environment, but originates from the design of the components themselves. This kind of emergent behaviour is similar to what has been observed for computational creative systems, and in particular for music improvisation [Bor05]. Consequently, we postulate that the dynamic reconfigurations of networks and music improvisations are governed by similar principles, and that both can be regarded as instances of more general phenomena that can be formalized over many-valued logics.

COMPONENTS OF SERVICE-ORIENTED SYSTEMS

We fix an arbitrary $\mathbb{RL}$-institution $\mathcal{I}$ that satisfies the following conditions:

1. the category of signatures $\mathbb{S}\mathrm{ig}^{\mathcal{I}}$ has designated pushouts and is equipped with a factorization system;
2. the truth-space functor preserves pullbacks as follows: $\mathrm{RL}^{\mathcal{I}}$ transforms any designated pushout $\langle \phi', i' \rangle$ of two signature morphisms ϕ and i into a pullback of morphisms of residuated lattices such that if $\mathrm{RL}^{\mathcal{I}}(i)$ is an identity, then $\mathrm{RL}^{\mathcal{I}}(i')$ is an identity too;

$$\begin{tikzcd}
\bullet \arrow[r, "i"] \arrow[d, "\phi"'] \arrow[dr, phantom, "po"] & \bullet \arrow[d, "\phi'"] \\
\bullet \arrow[r, "i'"'] & \bullet
\end{tikzcd}$$

3. the truth-space functor preserves and reflects factorizations. This means that, considering the factorization system $\langle \mathbb{E}, \mathbb{M} \rangle$ for $\mathbb{S}\mathrm{ig}^{\mathcal{I}}$, we have the following:

- if we apply the functor $\mathrm{RL}^{\mathcal{I}}$ to $\mathbb{E}$ and $\mathbb{M}$, we obtain the subcategories of $\mathbb{M}_{\mathbb{RL}}$ and $\mathbb{E}_{\mathbb{RL}}$ from Proposition 2.3.14. Note that, given the contravariant nature of $\mathrm{RL}^{\mathcal{I}}$, epimorphisms of $\mathcal{I}$-signatures are mapped to monomorphisms of residuated lattices, while monomorphisms are mapped to epimorphisms;
- for any factorization $e\,;m$ of a residuated-lattice morphism $\mathrm{RL}^{\mathcal{I}}(\phi)$, there exists a factorization $e_{\phi}\,;m_{\phi}$ of the signature morphism ϕ such that $\mathrm{RL}^{\mathcal{I}}(e_{\phi}) = m$ and $\mathrm{RL}^{\mathcal{I}}(m_{\phi}) = e$.

The conditions above hold for all the concrete examples of $\mathbb{RL}$-institutions that we have presented in this thesis. This is mainly due to the definition of the categories of signatures as products of the category $\mathbb{RL}^{\mathrm{op}}$ with other categories, and consequently, to the componentwise definition of constructions such as pushouts and factorization systems. In particular, for a soft-constraints institution $\mathrm{CSP}(\mathcal{J})$, it suffices that $\mathbb{S}\mathrm{ig}^{\mathcal{J}}$ has pushouts and admits a factorization system (see Propositions 3.2.15 and 3.2.18).

REMARK 4.1.1. The fact that we fix an $\mathbb{RL}$-institution is not a limitation per se. The framework that we propose could be used in a heterogeneous setting, meaning that the service components could be written over different many-valued institutions, connected through various comorphisms. In other words, one would have to consider an indexed $\mathbb{RL}$-institution, which can be flattened by means of a Grothendieck construction as in Section 3.1.

However, one would still have to ensure that the result of the construction enjoys the properties mentioned above. For this purpose, we recall from [TBG91] that if the category $\mathbb{I}$ of indexes is cocomplete, as well as the category of signatures of every 'local' $\mathbb{RL}$-institution in the indexed institution, and if the indexed signature functor $\mathbb{S}\mathrm{ig}\colon \mathbb{I}^{\mathrm{op}} \to \mathbb{C}\mathrm{at}$ is locally reversible (in the sense that, for every index morphism $u\colon i \to i'$ in $\mathbb{I}$, the functor $\mathbb{S}\mathrm{ig}(u)\colon \mathbb{S}\mathrm{ig}(i') \to \mathbb{S}\mathrm{ig}(i)$ has a left adjoint), then $\mathbb{S}\mathrm{ig}^{\#}$ is cocomplete.

One can also show, along the lines of [Dia11] that, under similar conditions to those presented above, the category of signatures obtained through the Grothendieck construction can be endowed with a factorization system.

Finally, it is easy to infer now that if the truth-space functors RL of every $\mathbb{RL}$-institution preserve pullbacks and preserves and reflects factorizations, and if $\mathbb{S}\mathrm{ig}$ is locally reversible, then the truth-space functor of the flattening will also preserve pullbacks and preserve and reflect factorizations.

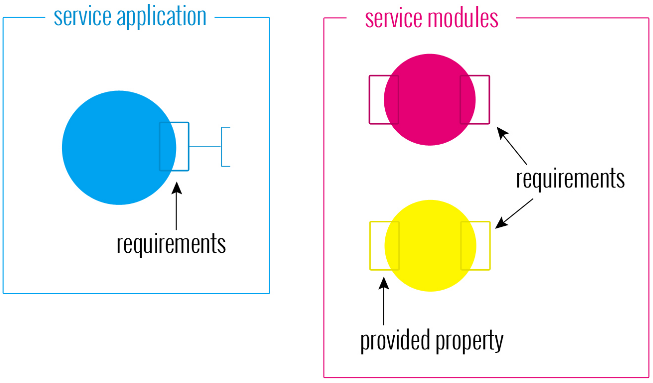


FIGURE 4.1. Components of service-oriented systems

In our framework for service-oriented computing, we consider two kinds of units, *service applications* and *service modules*, as in Figure 4.1. Service applications can be seen as units that require access to services or resources. They consist of an *orchestration* part, describing what the unit intends to do (the blue disc in Figure 4.1), and some *interfaces* describing the services required (the blue rectangle in Figure 4.1). In particular, interfaces are subspecifications of the given orchestration together with properties – logic sentences – that describe the preferences of the unit for a given quality of service.

DEFINITION 4.1.2 (Service application). A *service application* $\langle \Sigma, I, R \rangle$ consists of a signature $\Sigma \in |\mathrm{Sig}^{\mathcal{J}}|$, called *orchestration*, together with a finite family $I = \{i_x\}_{x \in \overline{1,n}}$ of *interfaces*[1], that is, a family of monic signature morphisms $i_x \colon \Sigma_x \to \Sigma$ such that $\mathrm{RL}^{\mathcal{J}}(i_x) = \mathrm{id}_{\mathrm{RL}^{\mathcal{J}}(\Sigma)}$, and their associated *requirements* $R = \{r_x \in \mathrm{Sen}^{\mathcal{J}}(\Sigma_x)\}_{x \in \overline{1,n}}$. We refer to a pair $\langle \Sigma_x, r_x \rangle$ consisting of the domain of an interface and its corresponding requirement as a *requires-specification*.

$$\begin{tikzcd}
 & \Sigma_1 \arrow[dl, "i_1"'] \arrow[r, mapsto] & r_1 \\
\Sigma & \vdots & \\
 & \Sigma_n \arrow[ul, "i_n"] \arrow[r, mapsto] & r_n
\end{tikzcd}$$

We illustrate each concept that we introduce in this section with an example related to the book-selling scenario from Section 3.2.1, written over the many-valued logic of CSP($\underline{\mathcal{FOL}}$).

EXAMPLE 4.1.3 (Books). As part of our running example, we consider a service application $\mathcal{C} = \langle \Sigma, I, R \rangle$ whose orchestration Σ is the presentation $\mathsf{Customer}$ from Figure 3.2, and whose single interface consists of the identity morphism and the requirement R given by the constraint sentence:

$$\mathsf{languagePref}(\mathsf{language}(\mathsf{book})) \wedge$$
$$\mathsf{deliveryPref}(\mathsf{book}, \mathsf{delivery}, \mathsf{deliveryTime}(\mathsf{book}, \mathsf{delivery})).$$

Service modules are like service applications but, in addition to requirements, they provide functionalities or resources. In this sense, they have an orchestration part (the pink and yellow discs in Figure 4.1), interfaces for the services required (similarly to service applications), as well as a provides interface (the pink and yellow rectangle depicted on the left side of the disks in Figure 4.1).

DEFINITION 4.1.4 (Service module). A *service module* $\langle \Omega, P, J, Q \rangle$ consists of an *orchestration* $\Omega \in |\mathrm{Sig}^{\mathcal{J}}|$, a *provides-property* $P \in \mathrm{Sen}^{\mathcal{J}}(\Omega)$, a finite family $J = \{j_y\}_{y \in \overline{1,m}}$ of *interfaces* $j_y \colon \Omega_y \to \Omega$, together with a family of associated *requirements* $Q = \{q_y \in \mathrm{Sen}^{\mathcal{J}}(\Omega_y)\}_{y \in \overline{1,m}}$.

$$\begin{tikzcd}
 & & \Omega_1 \arrow[dl, "j_1"'] \arrow[r, mapsto] & q_1 \\
P & \Omega \arrow[l, mapsto] & \vdots & \\
 & & \Omega_m \arrow[ul, "j_m"] \arrow[r, mapsto] & q_m
\end{tikzcd}$$

[1] Recall that by $\overline{1,n}$ we denote the set $\{1, \ldots, n\}$.

EXAMPLE 4.1.5 (Books). We define a service module $\mathcal{S} = \langle \Omega, P, J, Q \rangle$ for the application $\mathcal{C}$ given in Example 4.1.3 by taking Ω as the specification Supplier from Figure 4.2, the provides-property $P = \mathsf{available}(\mathsf{book}, \mathsf{delivery})$, and the requirement $Q = \mathsf{deliverable}(\mathsf{book}, \mathsf{delivery}, \mathsf{days})$ defined over Ω (i.e. J consists of an identity). The module guarantees the delivery of a book b for a method of delivery d within $\mathsf{deliveryTime}(b, d)$ days, but in turn depends on another external delivery-service provider.

```
logic FOL(ResiduatedLattices)
spec Supplier = BookData    then
    ops    deliveryTime : Book × Delivery ⟶ Nat
           available : Book × Delivery ⟶ L
           deliverable : Book × Delivery × Nat ⟶ L
    cvars  book : Book; delivery : Delivery; days : Nat

    ∀ b : Book; d : Delivery; n, n′ : Nat
    • deliverable(b, d, n) = 0 if n > deliveryTime(b, d)
    • deliverable(b, d, n) ≤ deliverable(b, d, n′) if n ≤ n′
    • deliverable(b, d, n) = 0 if (b, d) does not belong to the following table[2]
    • available(b, d) = 1 ⇔ (b, d) belongs to the following table,
                                  or 0 otherwise
```

id	book	language	delivery	deliveryTime
1.1	Schiele	de	standard	6
1.2			express	3
1.3			online	0

FIGURE 4.2. The presentation Supplier

DEFINITION 4.1.6 (ζ-satisfiability of an application). Given $\zeta \in \mathrm{RL}^{\mathcal{J}}(\Sigma)$, a *service application* $\langle \Sigma, I, R \rangle$ is *ζ-satisfiable* if all of its requirements can be satisfied at once with a value greater than ζ; intuitively, there exists a model of its orchestration that satisfies R with at least the value ζ: $\bigvee_{M \in |\mathrm{Mod}^{\mathcal{J}}(\Sigma)|} \big(\bigwedge_{x \in \overline{1,n}} M \models^{\mathcal{J}}_{\Sigma} i_x(r_x) \big) \geq \zeta$.

DEFINITION 4.1.7 (η-correctness of a service module). Given $\eta \in \mathrm{RL}^{\mathcal{J}}(\Omega)$ a *service module* $\mathcal{M} = \langle \Omega, P, J, Q \rangle$ is said to be *η-correct* if P is a consequence of Q with a value $\eta_{\mathcal{M}}$ greater than η. Formally, this means that $\eta_{\mathcal{M}} = \big(\{ j_y(q_y) \}_{y \in \overline{1,m}} \models^{\mathcal{J}}_{\Omega} P \big) \geq \eta$.

EXAMPLE 4.1.8 (Books). The module $\mathcal{S}$ from Example 4.1.5 is 1-correct: models with interpretations for book and delivery that do not belong to

[2] The table is only a convenient abbreviation for a set of sentences that specify, for example, that the book "Schiele" is available in German with 3-day express delivery. The column "id" is just an annotation that we use to reference the rows.

the given table satisfy neither P nor Q, while the models in which the interpretations are according to the table satisfy P with value 1.

DYNAMIC PROCESSES OF SERVICE-ORIENTED COMPUTING

We now focus on the execution of service applications in the context of a fixed set *Rep* of service modules – a *service repository*. Each execution step is triggered by the need to fulfil a requirement of the current application, which in the context of our work corresponds to a requires-specification. Similarly to conventional soft-constraint satisfaction problems, the goal is to maximize the satisfaction of the requirement. To this end, we distinguish three elementary processes: discovery, selection and binding.

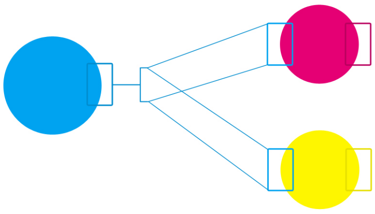


FIGURE 4.3. Service module discovery

SERVICE DISCOVERY. Let $\mathcal{A} = \langle \Sigma, I, R \rangle$ be a service application and $\langle \Sigma_k, r_k \rangle$ one of its requires-specifications. Unlike the selection and binding processes, we model the *discovery* of new service modules to be bound to $\mathcal{A}$ in a minimal way: all we assume is that it provides a set of possible matches (depicted as blue rectangles linking the interfaces of the service application with those of the service modules in Figure 4.3) – pairs $\langle \mathcal{M}, \phi \rangle$ of service modules $\mathcal{M} = \langle \Omega, P, J, Q \rangle$ from *Rep* and *attachment morphisms* $\phi \colon \Sigma_k \to \Omega$, for which $\mathrm{RL}^{\mathcal{I}}(\phi)$ is in $\mathbb{M}_{\mathbb{RL}}$. Note that the output of the discovery process only depends on the repository and the selected requires-specification, and not on the application itself.

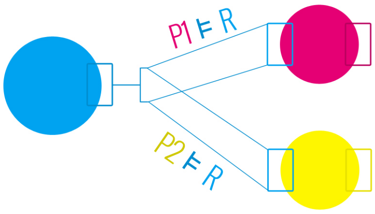


FIGURE 4.4. Service module selection

SERVICE SELECTION. In order to *select* from the set of discovered service modules the best module that satisfies the requirement, we compute, for each match $\langle \mathcal{M}, \phi \rangle$ provided by the discovery process, the compatibility score between the provides-property P guaranteed by the correctness of the service module $\mathcal{M}$ and of the requirement r_k of the application (in Figure 4.4, these are represented as annotations of the matches). To this end, we first compute the pushout $\langle i, j \rangle$ of the signature morphisms i_k and ϕ linking the requires-specification $\langle \Sigma_k, r_k \rangle$ to the orchestrations of the application and of the service module (see the diagram below), and then translate both the requirement and the provides-property to the vertex Σ' of the pushout:

$$\big(j(P) \vDash^{\mathcal{I}}_{\Sigma'} \mathrm{Sen}^{\mathcal{I}}(i_k ; i)(r_k)\big) = \bigwedge_{M \in |\mathrm{Mod}^{\mathcal{I}}(\Sigma')|} \big(M \vDash^{\mathcal{I}}_{\Sigma'} j(P)\big) \rightarrow \big(M \vDash^{\mathcal{I}}_{\Sigma'} \mathrm{Sen}^{\mathcal{I}}(i_k ; i)(r_k)\big).$$

For different service providers, these values would belong to different lattices (those defined by the providers), hence we have to further translate them to the lattice of the service application via the morphisms $\mathrm{RL}^{\mathcal{I}}(i)$ in order to be able to compare them. Here it is useful to note that $\mathrm{RL}^{\mathcal{I}}(i) = \mathrm{RL}^{\mathcal{I}}(\phi)$ because $\mathrm{RL}^{\mathcal{I}}(i_k)$ and $\mathrm{RL}^{\mathcal{I}}(j)$ are identities.

$$\begin{tikzcd}
\Sigma_k \arrow[r, "i_k"] \arrow[d, "\phi"'] \arrow[dr, phantom, "po"] & \Sigma \arrow[d, "i"] \arrow[dr, phantom, "="] & \Sigma_x \arrow[l, "i_x"'] \arrow[d, "e^\Sigma_x"] \\
\Omega \arrow[r, "j"'] & \Sigma' & \Sigma'_x \arrow[l, "m^\Sigma_x"] \\
\Omega_y \arrow[u, "j_y"] \arrow[r, "e^\Omega_y"'] \arrow[ur, phantom, "="] & \Omega'_y \arrow[u, "m^\Omega_y"'] &
\end{tikzcd}
\qquad
\begin{tikzcd}
& \Sigma'_x \arrow[dl, "m^\Sigma_x"'] \arrow[r, mapsto] & e^\Sigma_x(r_x) \\
\Sigma' & & \\
& \Omega'_y \arrow[ul, "m^\Omega_y"] \arrow[r, mapsto] & e^\Omega_y(q_y)
\end{tikzcd}$$

In the diagrams above, x is the index of a requirement of the application different from k, y is the index of a requirement of the selected module, $e^\Sigma_x ; m^\Sigma_x$ and $e^\Omega_y ; m^\Omega_y$ are the factorizations of the composed morphisms $i_x ; i$ and $j_y ; j$, respectively.

However, computing such compatibility scores is not enough: the selection of a best module for the distinguished requirement of the application must also take into account the correctness of the modules. Thus, for every match $\langle \mathcal{M}, \phi \rangle$, we have to multiply the score $\mathrm{RL}^{\mathcal{I}}(i)(j(P) \vDash^{\mathcal{I}}_{\Sigma'} \mathrm{Sen}^{\mathcal{I}}(i_k ; i)(r_k))$ obtained as above with $\mathrm{RL}^{\mathcal{I}}(\phi)(\eta_{\mathcal{M}})$, the correctness of $\mathcal{M}$. Finally, we select those service modules for which this product – computed according to the multiplication operation of $\mathrm{RL}^{\mathcal{I}}(\Sigma)$ – is maximal.

$$sel(Rep, \mathcal{A}, \Sigma_k, r_k) = \underset{\langle \mathcal{M}, \phi \rangle}{\arg\max}\{\mathrm{RL}^{\mathcal{I}}(\phi)(\eta_{\mathcal{M}}) * \mathrm{RL}^{\mathcal{I}}(i)\big(j(P) \models^{\mathcal{I}}_{\Sigma'} \mathrm{Sen}^{\mathcal{I}}(i_k\,;i)(r_k)\big)\}$$

EXAMPLE 4.1.9 (Books).

```
logic FOL(ResiduatedLattices)
spec Supplier2 = BookData    then
      ops    deliveryTime : Book × Delivery ⟶ Nat
             available : Book × Delivery ⟶ L
             deliverable : Book × Delivery × Nat ⟶ L
      cvars  book : Book; delivery : Delivery; days : Nat

      ∀ b : Book; d : Delivery; n, n′ : Nat
      • deliverable(b, d, n) = 0 if n > deliveryTime(b, d)
      • deliverable(b, d, n) ≤ deliverable(b, d, n′) if n ≤ n′
      • deliverable(b, d, n) = 0 if (b, d) does not belong to the following table
      • available(b, d) = 1 ⇔ (b, d) belongs to the following table,
                                 or 0 otherwise
```

id	book	language	delivery	deliveryTime
2.1	ChagallMaVie	fr	standard	14
2.2			express	8
3.1	Munch	en	standard	6
3.2			online	0

FIGURE 4.5. The presentation Supplier2

Consider the repository $Rep = \{\mathcal{S}, \mathcal{S}'\}$ containing the service module $\mathcal{S}$ from Example 4.1.5, and a new service module $\mathcal{S}' = \langle \Omega', P', J', Q' \rangle$ with Ω' as in Figure 4.2, $P' = P$, and $Q' = Q$. When selecting a best supplier for the service application $\mathcal{C}$ from Example 4.1.3, the books that best fit the preferences are the online version of "Schiele" (1.3) for $\mathcal{S}$ and "Chagall – Ma vie" with an express delivery (2.2) for $\mathcal{S}'$. In principle, we would need to compute the compatibility scores between Customer and Supplier on the one hand, and between Customer and Supplier2, on the other, using all possible models. However, due to the way the specifications are written, the choice of the best book for each supplier can be calculated directly from the axioms. First, the constraint variables book and delivery are limited to the interpretations defined by the tables. Second, the axioms of Customer that express specific preferences, such as for a language, make it feasible to determine the best books provided by each supplier, for any model. With respect to the language, Book 3 is the least preferred, while 2.1 and 2.2 are the most preferred because languagePref(en) ≤ languagePref(de) ≤ languagePref(fr). In order to determine the best buying option, it suffices now to decide which variant of 2.1 and 2.2 is the most suitable for our constraints, which we do by comparing their delivery options: since express delivery is preferred to standard when the latter does not guarantee a delivery within seven days, the best choice is 2.2. Because both service modules are correct with value 1, the selection process is not influenced by their correctness values, and $\mathcal{S}'$ is chosen as the best supplier.

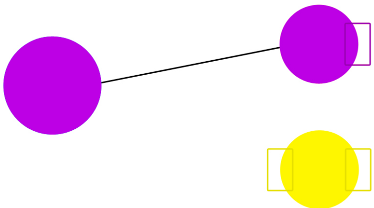


FIGURE 4.6. Service binding

SERVICE BINDING. After selecting a service module – non-deterministically from the set $sel(Rep, \mathcal{A}, \Sigma_k, r_k)$ –, the application will commit to the chosen provider through a *binding process* that changes the application as follows:

- The new orchestration is the vertex Σ' of the pushout $\langle i, j \rangle$; in Figure 4.6, the new orchestration is represented by the two purple discs.
- Apart from the interface i_k corresponding to the distinguished requirement, the interfaces of the application are preserved via a factorization of the composition of the old interfaces and the morphism of orchestrations i: for $x \in \overline{1,n} \setminus \{k\}$, we obtain the interface $m^{\Sigma}_x \colon \Sigma'_x \to \Sigma'$ by choosing a suitable factorization $e^{\Sigma}_x \,;\, m^{\Sigma}_x$ of the composed morphism $i_x \,;\, i$ as in Remark 4.1.10.
- The interface i_k is replaced by the interfaces of the selected service module: for $y \in \overline{1,m}$, $m^{\Omega}_y \colon \Omega'_y \to \Sigma'$ is the monic in the factorisation of $j_y \,;\, j$.
- The distinguished requirement r_k is replaced by the requirements of the selected module, i.e. $\{e^{\Omega}_y(q_y)\}_{y \in \overline{1,m}}$, while the other requirements of the application are kept: for $x \in \overline{1,n} \setminus \{k\}$, r_x is translated to $e^{\Sigma}_x(r_x)$.

REMARK 4.1.10. To ensure the well-definedness of service binding, we need to verify that we can choose $e^{\Sigma}_x \,;\, m^{\Sigma}_x$ such that $\mathrm{RL}^{\mathcal{I}}(m_x)$ is an identity. Consider the diagram below. Because $\mathrm{RL}^{\mathcal{I}}$ preserves pullbacks, the square on the right side is a pullback, and $\mathrm{RL}^{\mathcal{I}}(i_k) = \mathrm{RL}^{\mathcal{I}}(j)$ are identities. Moreover, because ϕ is an attachment morphism, we know that $\mathrm{RL}^{\mathcal{I}}(\phi)$ is a monic from $\mathbb{M}_{\mathbb{RL}}$. And because $\mathrm{RL}^{\mathcal{I}}(i) = \mathrm{RL}^{\mathcal{I}}(\phi)$, $\mathrm{RL}^{\mathcal{I}}(i)$ is also in $\mathbb{M}_{\mathbb{RL}}$.

$$\begin{tikzcd}
\Sigma_k \arrow[r, "i_k"] \arrow[d, "\phi"'] \arrow[dr, phantom, "po"] & \Sigma \arrow[d, "i"] \\
\Omega \arrow[r, "j"'] & \Sigma'
\end{tikzcd}
\qquad
\begin{tikzcd}
\mathrm{RL}^{\mathcal{J}}(\Sigma_k) & \mathrm{RL}^{\mathcal{J}}(\Sigma) \arrow[l, "\mathrm{RL}^{\mathcal{J}}(i_k)"'] \\
\mathrm{RL}^{\mathcal{J}}(\Omega) \arrow[u, "\mathrm{RL}^{\mathcal{J}}(\phi)"] & \mathrm{RL}^{\mathcal{J}}(\Sigma') \arrow[l, "\mathrm{RL}^{\mathcal{J}}(j)"] \arrow[u, "\mathrm{RL}^{\mathcal{J}}(i)"']
\end{tikzcd}$$

For any other morphism $i_x \colon \Sigma_x \to \Sigma$ as in the figure on page 88, $\mathrm{RL}^{\mathcal{J}}(i_x \,;\, i) = \mathrm{RL}^{\mathcal{J}}(i) \,;\, \mathrm{RL}^{\mathcal{J}}(i_x) = \mathrm{RL}^{\mathcal{J}}(i) \,;\, \mathrm{id}_{\mathrm{RL}^{\mathcal{J}}(\Sigma)} = \mathrm{RL}^{\mathcal{J}}(i)$. Because $\mathrm{RL}^{\mathcal{J}}(i)$ is also in $\mathbb{M}_{\mathbb{RL}}$, from the property of reflexion of factorizations of $\mathrm{RL}^{\mathcal{J}}$ we know that there exists a factorization $e_x \,;\, m_x$ of $i_x \,;\, i$ such that $\mathrm{RL}^{\mathcal{J}}(e_x) = \mathrm{RL}^{\mathcal{J}}(i)$ and $\mathrm{RL}^{\mathcal{J}}(m_x) = \mathrm{id}_{\mathrm{RL}^{\mathcal{J}}(\Sigma')}$.

PROPOSITION 4.1.11 (Correctness of service binding). *Let $\mathcal{M} = \langle \Omega, P, J, Q \rangle$ be an η-correct module that matches a service application $\mathcal{A} = \langle \Sigma, I, R \rangle$ through a morphism $\phi \colon \Sigma_k \to \Omega$. If the selection process guarantees that the compatibility score of the requirement r_k of $\mathcal{A}$ and the provides-property P of $\mathcal{M}$ is at least δ, and if the resulting application $\mathcal{A}' = \langle \Sigma', I', R' \rangle$ of their binding is ζ-satisfiable, then $\mathcal{A}$ is ξ-satisfiable with $\xi = \mathrm{RL}^{\mathcal{J}}(\phi)(\eta * \delta * \zeta)$*[3].

PROOF. The ζ-satisfiability of $\mathcal{A}'$ means that

$$\bigvee_{M' \in |\mathrm{Mod}^{\mathcal{J}}(\Sigma')|} \Big(\bigwedge_{x \in \overline{1,n} \setminus \{k\}} M' \models^{\mathcal{J}}_{\Sigma'} m^{\Sigma}_x(e^{\Sigma}_x(r_x)) \wedge \bigwedge_{y \in \overline{m}} M' \models^{\mathcal{J}}_{\Sigma'} m^{\Omega}_y(e^{\Omega}_y(q_y)) \Big) \geq \zeta.$$

We choose[4] a Σ'-model M^* such that

$$\bigwedge_{x \in \overline{1,n} \setminus \{k\}} M^* \models^{\mathcal{J}}_{\Sigma'} m^{\Sigma}_x(e^{\Sigma}_x(r_x)) \wedge \bigwedge_{y \in \overline{1,m}} M^* \models^{\mathcal{J}}_{\Sigma'} m^{\Omega}_y(e^{\Omega}_y(q_y)) \geq \zeta.$$

It follows that

$$\bigwedge_{x \in \overline{1,n} \setminus \{k\}} M^* \models^{\mathcal{J}}_{\Sigma'} m^{\Sigma}_x(e^{\Sigma}_x(r_x)) \geq \zeta,$$

[3] Note that η is a value from the lattice of Ω, while δ and ζ are from the lattice of Σ'. This means that ξ should actually be given by $\mathrm{RL}^{\mathcal{J}}(\phi)(\eta) * \mathrm{RL}^{\mathcal{J}}(i_k \,;\, i)(\delta * \zeta)$. However, since $\mathrm{RL}^{\mathcal{J}}(i_k) = \mathrm{id}_{\mathrm{RL}^{\mathcal{J}}(\Sigma)}$ and $\mathrm{RL}^{\mathcal{J}}(\phi) = \mathrm{RL}^{\mathcal{J}}(i)$, we can write ξ as $\mathrm{RL}^{\mathcal{J}}(\phi)(\eta * \delta * \zeta)$.

[4] The choice of such a model M^* raises no difficulties when the lattice corresponding to Σ' is a total order. However, in general, one should first ensure that the element ζ is join-irreducible, i.e. there is a value x in the set over which we compute the join that is greater or equal to ζ. In this situation, we pick M^* as one of the models corresponding to x.

and because $m_x^\Sigma(e_x^\Sigma(r_x)) = i(i_x(r_x))$ for $x \in \overline{1,n} \setminus \{k\}$, we have

$$\bigwedge_{x\in\overline{1,n}\setminus\{k\}} M^* \vDash_{\Sigma'}^{\mathcal{I}} i(i_x(r_x)) \geq \zeta.$$

From the satisfaction condition for the signature morphism i we know that $M^*\restriction_i \vDash_\Sigma^{\mathcal{I}} i_x(r_x) = \mathrm{RL}^{\mathcal{I}}(i)(M^* \vDash_{\Sigma'}^{\mathcal{I}} i(i_x(r_x)))$ for $x \in \overline{1,n} \setminus \{k\}$, and thus

$$\bigwedge_{x\in\overline{1,n}\setminus\{k\}} M^*\restriction_i \vDash_\Sigma^{\mathcal{I}} i_x(r_x) \geq \mathrm{RL}^{\mathcal{I}}(i)(\zeta).$$

Similarly, we can deduce from the ζ-satisfiability of $\mathcal{A}'$ that

$$\bigwedge_{y\in\overline{1,m}} M^*\restriction_j \vDash_\Omega^{\mathcal{I}} j_y(q_y) \geq \mathrm{RL}^{\mathcal{I}}(j)(\zeta) = \mathrm{id}_{\Sigma'}(\zeta) = \zeta. \tag{4.1}$$

From the η-correctness of $\mathcal{M}$ we know

$$(M^*\restriction_j \vDash_\Omega^{\mathcal{I}} \{j_y(q_y)\}_{y\in\overline{1,m}}) \to (M^*\restriction_j \vDash_\Omega^{\mathcal{I}} P) \geq \eta. \tag{4.2}$$

Because the selection process guaranteed a compatibility score of at least $\eta * \mathrm{RL}^{\mathcal{I}}(j)(\delta) = \eta * \delta$ between $\mathcal{A}$ and $\mathcal{M}$, we know that

$$\bigwedge_{M'\in|\mathrm{Mod}^{\mathcal{I}}(\Sigma')|} (M' \vDash_{\Sigma'}^{\mathcal{I}} j(P)) \to (M' \vDash_{\Sigma'}^{\mathcal{I}} j(\phi(r_k))) \geq \delta,$$

and thus $(M^* \vDash_{\Sigma'}^{\mathcal{I}} j(P)) \to (M^* \vDash_{\Sigma'}^{\mathcal{I}} j(\phi(r_k))) \geq \delta$. From the satisfaction condition for j, it follows that

$$(M^*\restriction_j \vDash_\Omega^{\mathcal{I}} P) \to (M^*\restriction_j \vDash_\Omega^{\mathcal{I}} \phi(r_k)) \geq \mathrm{RL}^{\mathcal{I}}(j)(\delta) = \delta. \tag{4.3}$$

From (4.2), (4.3) and Proposition 2.3.12, it follows that

$$(M^*\restriction_j \vDash_\Omega^{\mathcal{I}} \{j_y(q_y)\}_{y\in\overline{1,m}}) \to (M^*\restriction_j \vDash_\Omega^{\mathcal{I}} \phi(r_k)) \geq \eta * \delta. \tag{4.4}$$

By applying Proposition 2.3.11 to (4.1) and (4.4), it follows that

$$M^*\restriction_j \vDash_\Omega^{\mathcal{I}} \phi(r_k) \geq \eta * \delta * \zeta,$$

and from the satisfaction condition for ϕ we know that $M^*\restriction_j\restriction_\phi \vDash_{\Sigma_k}^{\mathcal{I}} r_k \geq \mathrm{RL}^{\mathcal{I}}(\phi)(\eta * \delta * \zeta)$. We finally obtain that $M^*\restriction_i \vDash_\Sigma^{\mathcal{I}} i_k(r_k) \geq \mathrm{RL}^{\mathcal{I}}(\phi)(\eta * \delta * \zeta)$, since $M^*\restriction_i\restriction_{i_k} = M^*\restriction_j\restriction_\phi$ and $\mathrm{RL}^{\mathcal{I}}(i_k) = id_{\mathrm{RL}^{\mathcal{I}}(\Sigma)}$. All we have to do now is to compare the satisfiability of the requirements r_k and r_x, for $x \in \overline{1,n} \setminus \{k\}$.

Because $\mathrm{RL}^{\mathcal{I}}(i) = \mathrm{RL}^{\mathcal{I}}(\phi)$, and $\zeta \geq \eta * \delta * \zeta$, we have that $\mathrm{RL}^{\mathcal{I}}(i)(\zeta) \geq \mathrm{RL}^{\mathcal{I}}(\phi)(\eta * \delta * \zeta)$, and hence $\mathcal{A}$ is proved to be $\mathrm{RL}^{\mathcal{I}}(\phi)(\eta * \delta * \zeta)$-satisfiable. □

FUNCTIONAL REQUIREMENTS. As mentioned in the beginning of the chapter, the formalization of service-oriented architectures that we present in this thesis focuses on non-functional requirements (quality-of-service constraints) of service applications and modules. That is not to say, however, that it cannot accommodate as well functional requirements, i.e., properties describing the actual functionality of external services that a component needs to discover and bind to in order to fulfil its business goal. For the purpose of this study, a way in which one could capture functional requirements is by writing specifications over a Boolean logic that is expressive enough to describe the manner in which service modules and applications are expected to work. For instance, in Srml, the Sensoria Reference Modelling Language [Fia+11], functional requirements are captured by specifications written in a temporal logic of stateful interactions [FLA12]. Here, the key property of functional requirements is the fact that their evaluation is Boolean (either 0 or 1), regardless of the lattice in which the evaluation takes place. We envisage two ways in which this could be achieved:

- A first solution would imply choosing an $\mathbb{RL}$-institution that is expressive enough to capture non-functional requirements, as well as functional requirements as necessarily true statements. That is, for every sentence ρ there exists a sentence γ – whose intuitive meaning is that ρ is necessarily true – that is evaluated to 1 (in any model) when ρ is evaluated to 1, and to 0 when ρ is evaluated to any other value than 1. For example, in a logic with negation and implication and with the underlying truth structure given by the Łukasiewicz residuated lattice $\mathcal{L}_3$, for any sentence ρ, the sentence $\neg(\rho \rightarrow \neg\rho)$, which corresponds to the Łukasiewicz necessitation [Mal93], is a functional requirement.
- Another possible solution would be using a combination of two logics: a Boolean logic – like the temporal one used in Srml – for describing functional requirements, and a many-valued logic for capturing non-functional requirements as soft constraints. This combination of logics can be obtained by means of a (categorical) product of institutions as described in [ST12, Chapter 4].

In both cases, the service discovery process would restrict the service repository to the modules that satisfy the functional requirements with value 1, while the selection process would single out those service modules that satisfy the soft constraints with the highest values.

4.2 CLAPPING MUSIC

To highlight the connection between creative systems and service-oriented computing, let us now focus on an example of a music-improvisation process described in terms of service discovery, selection, and binding of components written over the logic $\mathcal{BGP}$ from Section 3.3.

We build on the Clap! example (3.3.1) from Section 3.3, and discuss Steve Reich's minimalist composition "Clapping Music" written in 1972 [Rei80]. Although it is a complete composition, with no musical segments meant to be improvised, "Clapping Music" constitutes a good reference for our purpose due to its clarity. This simplification is intended to: (1) underline the fact that the freeness of the performance does not reside primarily in the qualitative aspects of the resulting music, but in the nature of the musicking process itself; and (2) alleviate the effort of the reader unfamiliar with basic notions of music theory. Written around a basic pattern very similar to the standard African 12/8 bell pattern, the piece should be played by two performers: one should continuously and unvaryingly repeat the basic pattern a in Figure 4.7 while the other should repeat and shift the pattern with one note after each eight bars [Rei74].

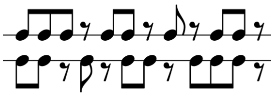


FIGURE 4.7. Basic pattern a of "Clapping Music"

We consider the universe of musical fragments to be the set of score fragments that can be obtained from composing the basic pattern a and the patterns obtained by shifting it, together with the prefixes of these shifts. We use fragments of this composition to exemplify specifications of incomplete musical segments written in Anthony Braxton's notation: we start from the first bar of the piece, the basic pattern a, and we let the performance develop according to both fixed, rigid instructions, and loose guidelines, which are subject to improvisation.

In what follows, we fix the set of music fragments MF to be the entire universe of fragments as described above, for every $\mathcal{BGP}$-signature $\langle \Sigma, E \rangle$. For simplicity, let us assume that the interpretation of these music fragments is also fixed: for every model M, we define the set $|M|$ as in Example 3.3.4. Because the interpretation of a music fragment is the same in all models, we make no distinction between the syntactical notation of a fragment and its interpretation. Furthermore, we describe the sequence M_{seq} of a $\langle \Sigma, E \rangle$-model M through regular-expression-like strings in which the $*$ symbol marks the fact that the sequence is open and admits any possible succession of sounds, while interrogation points mark the parts that are yet to be fixed – that is, those parts that are covered by formation symbols. These expressions correspond to sentences in E of the form $m@p$, with $m \in \mathrm{MF}$ and $p \in \mathtt{Nat}$, that fix elements of the sequence of sounds to be played.

SERVICE APPLICATION. Figure 4.8 specifies the starting score of "Clapping Music". Bar a is followed by a shape formation 🗀 specifying that the pattern should be repeated, but in a transformed state reminding of descending steps (fragment x), and by the fixed fragment b, to which other fragments may be added. This score can be formalized as a service application $\mathcal{A} = \langle \Sigma, I, R \rangle$ with:

- the orchestration Σ given by $\langle \mathcal{L}_\Sigma, \mathrm{MF}, \{$🗀$\}, E_\Sigma \rangle$, where E_Σ is the set of sentences $\{a@0, b@2\}$ that describe sequences of sounds of the form $a?b*$;
- a single interface $i_1 \colon \Sigma_1 \to \Sigma$ from $\Sigma_1 = \langle \mathcal{L}_\Sigma, \mathrm{MF}, \{$🗀$\}, \emptyset \rangle$, where i_1^ℓ, i_1^{mf} and i_1^{fs} are all identities, and $i_1^d = 1$ (to indicate that the variable fragment x appears at position 1). Notice that $E_{\Sigma'}$ is the empty set, meaning that the Σ'-models are not restricted to have any particular sequence of sounds, or, in other words, they correspond to all sequences of sounds, which are described as $*$. We choose not to limit the class of models through the interface because we want to be able to consider as a candidate in the selection process any music fragment satisfying the requirement R;
- the requirement $R =$ 🗀$_0(x)$ or, equivalently, 🗀$@0 \wedge (\sim x@0)$.

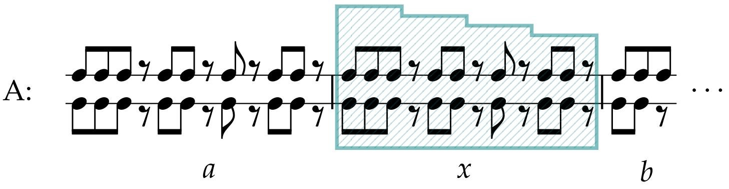


FIGURE 4.8. Score A of "Clapping Music"

SERVICE MODULES. To refine and continue the given music score A, we consider the scores B and C in Figure 4.9 as possible candidates. Formally, they are service modules $\mathcal{B} = \langle \Omega_\mathcal{B}, P_\mathcal{B}, J_\mathcal{B}, \emptyset \rangle$ and $\mathcal{C} = \langle \Omega_\mathcal{C}, P_\mathcal{C}, J_\mathcal{C}, Q_\mathcal{C} \rangle$ as follows:

- their orchestrations are $\Omega_\mathcal{B} = \langle \mathcal{L}_{\Omega_\mathcal{B}}, \mathrm{MF}, \emptyset, E_{\Omega_\mathcal{B}} \rangle$ and $\Omega_\mathcal{C} = \langle \mathcal{L}_{\Omega_\mathcal{C}}, \mathrm{MF}, \{\lhd\}, E_{\Omega_\mathcal{C}} \rangle$ with $E_{\Omega_\mathcal{B}}$ and $E_{\Omega_\mathcal{C}}$ restricting the models to those corresponding to the sound sequences $cbe*$ and $bebe?*$, respectively;

- they guarantee to begin with the fragments c[5] and $bebe$[6] through the provides-properties $P_{\mathcal{B}} = c@0$ and $P_{\mathcal{C}} = bebe@0$;
- we model the fact that the score B is completely fixed, not demanding improvisation, by considering the set of requirements to be empty;
- the interface $j_{\mathcal{C}}\colon \Omega'_{\mathcal{C}} \to \Omega_{\mathcal{C}}$ of $\mathcal{C}$ is defined similarly to the interface of $\mathcal{A}$: it consists of identities for the residuated lattice, the music-fragment space and the set of formation symbols, the natural number 1, indicating the position of the formation symbol in the score, and the set of sentences $E_{\Omega_{\mathcal{C}}}$ does not restrict the class of models, allowing all possible sequences of sounds;
- the requirement $Q_{\mathcal{C}} = \lhd_0(y)$.

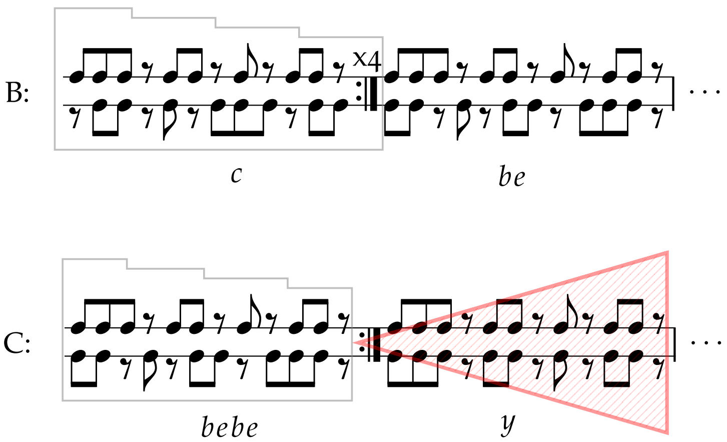


FIGURE 4.9. Scores B and C of "Clapping Music"

AD-HOC COMPOSITION. To continue the score of "Clapping Music", we consider a round of processes of discovery, selection and binding of other music fragments to the original fragment. Let the result of the discovery process be the set given by the modules $\mathcal{B}$ and $\mathcal{C}$. At this stage, because both the measure of similarity and formation symbols are so loosely interpreted, it is difficult to perform a selection that fits the actual practice and intuition of music improvisation. One way to address this is to fix at a semantic level a set of measures of similarity between music fragments and of acceptable interpretations for the formation symbol 🗀. How similar are the fragments c and $bebe$ to x, and how much can they be perceived as sounds in the shape of 🗀? This could be done, for example, by adding further sentences

[5] Note the ×4 superscript in Figure 4.9 B, denoting that c is the repetition of the highlighted fragment four times.
[6] $bebe$ is a single music fragment. Its name was thus chosen to highlight the fact that it starts with the prefix b.

to the orchestrations. We have already taken this route in Example 3.3.4, so we won't discuss this in further detail in order to allow us to concentrate on binding. Let us suppose that the score B is the result of the selection process, and focus on the binding of modules as a process of blending music fragments.

By computing the pushout of the morphisms i_1 and ϕ that map the requires-specification to the orchestration of the application and to the provides-specification of the service module, we amalgamate the musical sequences $a?b*$ and $cbe*$ of the models in $\mathrm{Mod}(\Sigma)$ and $\mathrm{Mod}(\Omega_{\mathcal{B}})$, obtaining the contiguous sound $acbe*$. Note the role of the delay 1 corresponding to the morphisms i_1 and j.

$$\begin{tikzcd}
\Sigma_1 \arrow[r, "i_1"] \arrow[d, "\phi"'] \arrow[dr, phantom, "po"] & \Sigma \arrow[d, "i"] \\
\Omega_{\mathcal{B}} \arrow[r, "j"'] & \Sigma'
\end{tikzcd}
\qquad
\begin{tikzcd}
* \arrow[r, "1"] \arrow[d, "0"'] & a?b* \arrow[d, "0"] \\
cbe* \arrow[r, "1"'] & acbe*
\end{tikzcd}$$

The substitution of the orchestration Σ of $\mathcal{A}$ with the vertex Σ' of the pushout will hence determine a refinement of the class of models $\mathrm{Mod}^{\mathcal{BGP}}(\Sigma)$. The effect of this refinement is twofold: one the one hand, it fills the improvisational gaps of the performance and, on the other hand, it moulds/gives shape to its evolution.

4.3 HISTORY AND VALUE SYSTEMS

In what follows, we analyse two distinguishing features of our method of selecting a best service module: unlike previous Boolean approaches [FL13a; FL13b], the procedure relies on arbitrary residuated lattices that may change through binding; moreover, it takes into account not only the properties of the supplier, but also the information encoded in the orchestration of the application. Each of these features raises new challenges in predicting which service module will be bound to the application.

HISTORY MATTERS. The choice of a best supplier usually depends on the orchestration of the application to be reconfigured. That orchestration, which we intuitively regard as the history of the execution of the application, is formed through the accumulation of the orchestrations of service modules that have been previously bound to the application. In music improvisation, for example, the music that has been performed has a great influence in the choice of the segment to be played next. However, there are situations in which the requirement that triggers a reconfiguration has such a rich meaning that we can even infer the characteristics of the music already played; the choice of the next fragment depends on the requirement alone. We identify below those situations in which the information contained by the orchestration of a service application becomes irrelevant to the selection of a best service module.

EXAMPLE 4.3.1 (Books). Consider the service application $\mathcal{C}' = \langle \Sigma', I', R \rangle$ with the orchestration Σ' defined as the presentation Customer of the application $\mathcal{C}$ from Example 4.1.3 to which we add the sentence

$\forall\, b\colon \mathsf{Book}, d\colon \mathsf{Delivery}, n\colon \mathsf{Nat}$
$\bullet\ \mathsf{deliveryPref}(d, b, n) = 0 \text{ if } n > 7$

and that has the same requirement as $\mathcal{C}$:

$$\begin{aligned} R = {} & \mathsf{languagePref}(\mathsf{language}(\mathsf{book})) \wedge \\ & \quad \mathsf{deliveryPref}(\mathsf{book}, \mathsf{delivery}, \mathsf{deliveryTime}(\mathsf{book}, \mathsf{delivery})). \end{aligned}$$

If we repeat the selection process for $\mathcal{C}'$ and the repository $\mathit{Rep} = \{\mathcal{S}, \mathcal{S}'\}$ as in Example 4.1.9, the supplier $\mathcal{S}$ will be chosen instead of $\mathcal{S}'$. This is due to the fact that the delivery time for Book 2 is greater than seven days, and thus it does not meet the time-limit imposed by the new application. This example illustrates how the change of the orchestration influences the selection process.

PROPOSITION 4.3.2. *Let $\mathcal{A} = \langle \Sigma, I, R \rangle$ be a service application and $\langle \Sigma_k, r_k \rangle$ a requires-specification written over an $\mathbb{RL}$-institution $\mathcal{I}$ having the model-amalgamation property. If the interface $i_k\colon \Sigma_k \to \Sigma \in I$ is a signature morphism that admits model expansions (i.e. if $\mathrm{Mod}^{\mathcal{I}}(i_k)$ is surjective on objects), the compatibility score between the requirement r_k of $\mathcal{A}$ and the provides-property of a service module $\mathcal{M} = \langle \Omega, P, J, Q \rangle$ can be evaluated directly with respect to the orchestration Ω of $\mathcal{M}$, rather than having to first compute the pushout of the application and the module.*

PROOF. The evaluation of the compatibility between the provides-property P and the requirement r_k with respect to the models of the module orchestration Ω reduces to computing

$$\mathrm{RL}^{\mathcal{I}}(\phi)\Big(\bigwedge_{M \in |\mathrm{Mod}^{\mathcal{I}}(\Omega)|} (M \vDash^{\mathcal{I}}_{\Omega} P) \rightarrow (M \vDash^{\mathcal{I}}_{\Omega} \phi(r_k)) \Big) \tag{4.5}$$

while the evaluation of the compatibility with respect to the vertex Σ' of the pushout of i_k and ϕ reduces to computing

$$\bigwedge_{M' \in |\mathrm{Mod}^{\mathcal{I}}(\Sigma')|} (M' \models^{\mathcal{I}}_{\Sigma'} j(P)) \to (M' \models^{\mathcal{I}}_{\Sigma'} (\phi\, ; j)(r_k)) \tag{4.6}$$

From the satisfaction condition for the morphism $j \colon \Omega \to \Sigma'$, we have $(M' \upharpoonright_j \models^{\mathcal{I}}_{\Omega} P) = \mathrm{RL}^{\mathcal{I}}(j)(M' \models^{\mathcal{I}}_{\Sigma'} j(P)) = (M' \models^{\mathcal{I}}_{\Sigma'} j(P))$ and $(M' \upharpoonright_j \models^{\mathcal{I}}_{\Omega} \phi(r_k)) = \mathrm{RL}^{\mathcal{I}}(j)(M' \models^{\mathcal{I}}_{\Sigma'} (\phi\, ; j)(r_k)) = (M' \models^{\mathcal{I}}_{\Sigma'} (\phi\, ; j)(r_k))$, for any Σ'-model M[7]. Hence, (4.6) can be written as

$$\mathrm{RL}^{\mathcal{I}}(\phi)(\bigwedge_{M' \in |\mathrm{Mod}^{\mathcal{I}}(\Sigma')|} (M' \upharpoonright_j \models^{\mathcal{I}}_{\Omega} P) \to (M' \upharpoonright_j \models^{\mathcal{I}}_{\Omega} \phi(r_k)) \tag{4.7}$$

We recall that in any weak amalgamation square

$$\begin{tikzcd}
\Sigma \arrow[r, "\varphi_1"] \arrow[d, "\varphi_2"'] & \Sigma_1 \arrow[d, "\theta_1"] \\
\Sigma_2 \arrow[r, "\theta_2"'] & \Sigma'
\end{tikzcd}$$

the morphism θ_2 admits model expansions whenever φ_1 has this property. Applying this general result to the pushout square used in computing the compatibility of $\mathcal{A}$ and $\mathcal{M}$, it follows that, since i_k admits model expansions, j also admits model expansions, hence (4.7) equals (4.5). □

The following result captures the conditions in which a signature morphism admits model expansions for the particular case of CSP($\mathcal{I}$) institutions.

PROPOSITION 4.3.3. *For a* CSP($\mathcal{I}$) *institution having the model-amalgamation property, a constraint signature morphism* $\langle \ell, \varphi \rangle \colon \langle \mathcal{L}, \Delta \rangle \to \langle \mathcal{L}', \Delta' \rangle$ *in* $\mathbb{S}\mathrm{ig}^{\mathrm{CSP}(\mathcal{I})}$*, with the underlying morphisms* $\varphi \colon \langle \Sigma, E \rangle \to \langle \Sigma', E' \rangle$ *and* $\ell \colon \mathcal{L}' \to \mathcal{L}$*, admits model expansions whenever* φ *admits model expansions and the reduct* $M \upharpoonright_\Delta$ *of any* $\langle \Sigma, E \rangle$*-model* M *is projective with respect to* ℓ*.*

PROOF. We have to prove that for any $\langle \mathcal{L}, \Delta \rangle$-model $\langle M, f \colon M \upharpoonright_\Delta \to \mathcal{L} \rangle$, there exists a $\langle \mathcal{L}', \Delta' \rangle$-model $\langle M', f' \colon M' \upharpoonright_{\Delta'} \to \mathcal{L}' \rangle$, such that its reduct $\langle M' \upharpoonright_\varphi, f'\, ; \ell \rangle$ along the morphism $\langle \ell, \varphi \rangle$ is equal to $\langle M, f \rangle$. Because $\varphi \colon \langle \Sigma, E \rangle \to \langle \Sigma', E' \rangle$ admits model expansions, it follows that for any $\langle \Sigma, E \rangle$-model M there exists a $\langle \Sigma', E' \rangle$-model M' such that $M' \upharpoonright_\varphi = M$, and thus we only have

[7] We recall that $\mathrm{RL}^{\mathcal{I}}(j)$ is an identity.

to check if there exists a morphism of residuated lattices $f' \colon M'{\upharpoonright}_{\Delta'} \to \mathcal{L}'$ such that $(f' \mathbin{;} \ell \colon M'{\upharpoonright}_{\Delta'} \to \mathcal{L}) = (f \colon M{\upharpoonright}_{\Delta} \to \mathcal{L})$.

$$\begin{tikzcd}
\mathcal{L}' \arrow[rr, "\ell"] & & \mathcal{L} \\
& M{\upharpoonright}_{\Delta} \arrow[ul, dashed, "f'"] \arrow[ur, "f"'] &
\end{tikzcd}$$

The existence of f' follows immediately from the projectivity of $M{\upharpoonright}_{\Delta}$ with respect to ℓ, and $M'{\upharpoonright}_{\Delta'} = (M'{\upharpoonright}_{\varphi}){\upharpoonright}_{\Delta} = M{\upharpoonright}_{\Delta}$. □

TRUTH MATTERS. The choice of a residuated lattice affects both the compatibility score (between a requirement and a provides-property) and the correctness of a service module. This matches the intuition that the system of values of the musicians involved in an improvisation process plays a crucial role for the actual improvisation part or characteristic of the performance.

EXAMPLE 4.3.4. Consider once again the service application $\mathcal{C}$ from Example 4.1.3 and two suppliers $\mathcal{S}_1$ and $\mathcal{S}_2$ whose orchestrations have the same underlying presentation – SimpleSupplier as in Figure 4.10. Moreover, they have the same provides-property,

$$\begin{aligned} P_1 = \mathsf{available}(\mathsf{book}, \mathsf{delivery}) \wedge (\mathsf{available}(\mathsf{book}, \mathsf{delivery}) \to \\ \mathsf{deliverable}(\mathsf{book}, \mathsf{delivery}, \mathsf{deliveryTime}(\mathsf{book}, \mathsf{delivery}))) \end{aligned}$$

and no requirements. The residuated lattices of the orchestrations of $\mathcal{S}_1$ and $\mathcal{S}_2$ differ: $\mathcal{S}_1$ is based on the Heyting algebra $\mathcal{HA}_{[0,1]}$ from Example 2.3.6, while $\mathcal{C}$ and $\mathcal{S}_2$ are based on the real-valued Łukasiewicz lattice $\mathcal{L}_{[0,1]}$ from Example 2.3.8.

The compatibility scores between the requirement

$$R = \mathsf{deliveryTime}(\mathsf{book}, \mathsf{delivery}, \mathsf{deliveryTime}(\mathsf{book}, \mathsf{delivery}))$$

of the service application $\mathcal{C}$ and the provides-property P_1 of $\mathcal{S}_1$ and $\mathcal{S}_2$ are 1 and 0.5, respectively[8]. Consequently, the selection process only determines $\mathcal{S}_1$ as a best service module. Notice that, even when $\mathcal{S}_1$ and $\mathcal{S}_2$ have the same underlying residuated lattices, the selection process may still depend on the matches between $\mathcal{C}$ and the two modules.

[8] The value 0.5 is obtained as a consequence of the form of the sentence P_1. We will discuss a more general situation in the next chapter.

```
logic FOL(ResiduatedLattices)
spec SimpleSupplier = BookData    then
      ops    deliveryTime : Book × Delivery ⟶ Nat
             available : Book × Delivery ⟶ L
             deliverable : Book × Delivery × Nat ⟶ L
```

FIGURE 4.10. The specification SimpleSupplier

Similarly, the correctness of a service module depends on its associated lattice.

EXAMPLE 4.3.5. Let $\mathcal{S}_3$ be a service module based on the extension of the presentation SimpleSupplier with the sentence

$$\forall\, b : \mathsf{Book}, d : \mathsf{Delivery}$$
$$\bullet\ \mathsf{deliverable}(b, d, \mathsf{deliveryTime}(b, d)) \to \mathsf{available}(b, d) = 1$$

Its provides-property is $P = \mathsf{available}(\mathsf{book}, \mathsf{delivery})$, and it has only one requirement, deliverable(book, delivery, deliveryTime(book, delivery)). The correctness of the module $\mathcal{S}_3$ depends on the residuated lattice of its orchestration: for any Heyting algebra, the module is correct with the value 1, while for the real-valued Łukasiewicz lattice $\mathcal{L}_{[0,1]}$, the module is only 0.5-correct. Of course, these values cannot be compared, as they belong to different lattices; still, the first one is absolute, while the second is not.

5

MANY-VALUED LOGIC PROGRAMMING

In this chapter, we show how computational-creativity aspects pertaining the service-oriented approach that we just explored can be simulated using a many-valued logic-programming framework based on arbitrary $\mathbb{RL}$-institutions. We aim towards a framework that is general enough to accommodate not only the service-oriented model presented in Chapter 4, but other possible types of computational creative systems as well. To this end, we abstract over service modules and applications as clauses and queries, and we model the fulfilment of the need for an external resource of an application as the execution of a logic program written over a many-valued institution. In the first section, we define the concept of solution to a problem and prove one of Herbrand's theorems, corresponding to the soundness of solutions. In the second section, we focus on the operational semantics, and define unification and resolution as abstractizations of service discovery and binding. Finally, we place our framework on the very rich map of multi-valued approaches to logic-programming, and present some conditions under which our system can be reduced to a classical, Boolean one, without loss of information.

Over the last decades, the traditional variant of logic programming based on Horn-clause logic has been redefined or adapted for a variety of other formalisms, including first-order and higher-order equational logic, hidden algebra, and constraint logic. These approaches have been the starting point for the generalization of logic-programming to arbitrary logics [Dia04; ŢF17] – a key process that enabled the development of the paradigm over a wide range of non-classical formalisms. Orthogonally, logic programming has been extended to a many-valued setting for reasoning under uncertainty, inconsistency, vagueness and preferences, resulting in several fuzzy [Voj01], probabilistic and possibilistic [NS92; Luk98; Luk99] variants of the classical Horn-clause logic programming [Sha83; KLV04; Bes+17b].

However, the concept of graded consequence has hitherto received little attention in the logic-programming community. While the name of the field, *many-valued logic programming*, suggests that the core elements of logic programming are dealing with uncertainty, we argue that this happens only at a syntactical, operational level, while the denotational semantics of these approaches is in essence Boolean. In what follows, we study the ramifications of considering graded entailment as a main ingredient of many-valued logic-programming. We start with institution-independent abstractions of the logic-programming concepts of clause, query, logic program, and solution. These abstractions allow us to look beyond the operational semantics of many-valued logic programming – the focus in the literature until now – and to further investigate the relationship between the denotational semantics and the notions of unification, computed solution, and resolution in a many-valued context.

The motivation behind this investigation comes from the need to liberate our general setting for creative systems from the particular restrictions that come with the model of service-oriented computing discussed in Chapter 4. In other words, by abstracting to a general logic-programming framework, we can define the main actors playing the roles of clauses and queries in a different way, not necessarily as service modules and applications, while keeping the core of the mechanisms of unification and resolution.

5.1 DENOTATIONAL SEMANTICS

A LOGICAL VIEW ON SERVICE-ORIENTED PROCESSES

In [ŢF15b], the authors presented how the denotational and the operational semantics of standard logic programming can be generalized to address both the static and the dynamic facets of service-oriented computing, resulting in a new, service-based version of the logic-programming paradigm. The study was developed starting from an analogy between the discovery of a service module to be bound to a service application and the search for a clause in order to solve a query. Although different in many aspects from our model of service-oriented computing, including the Boolean nature of the selection and binding processes, this variant of logic programming for services serves as inspiration for our approach. Intuitively, we capture service applications as queries, service modules as clauses, service-discovery process as unification, service selection as a method for choosing a best unifier, and service binding as resolution.

To this end, we define a many-valued logic for specifying interfaces of services over an arbitrary $\mathbb{RL}$-institution. For the examples discussed in the previous chapter, one needs to consider such a logic over CSP($\mathcal{FOL}$) or $\mathcal{BGP}$.

THE $\mathbb{RL}$-INSTITUTION OF MANY-VALUED $\mathcal{I}$-INTERFACES.

DEFINITION 5.1.1. Let $\mathcal{I}$ be an $\mathbb{RL}$-institution for which the category of signatures is equipped with a (kind of) factorization system such that any signature morphism has a unique factorization[1]. We define $\mathrm{INT}(\mathcal{I})$ as the tuple $\langle \mathbb{S}\mathrm{ig}^{\mathrm{INT}}, \mathrm{Sen}^{\mathrm{INT}}, \mathrm{Mod}^{\mathrm{INT}}, \mathrm{RL}^{\mathrm{INT}}, \models^{\mathrm{INT}} \rangle$ where:

- $\mathbb{S}\mathrm{ig}^{\mathrm{INT}} = \mathbb{S}\mathrm{ig}^{\mathcal{I}}$,
- $\mathrm{Mod}^{\mathrm{INT}} = \mathrm{Mod}^{\mathcal{I}}$,
- $\mathrm{RL}^{\mathrm{INT}} = \mathrm{RL}^{\mathcal{I}}$,
- For every signature Σ, the elements of $\mathrm{Sen}^{\mathrm{INT}}(\Sigma)$ are pairs $\langle i_r, r \rangle$ consisting of a monic signature morphism[2] $i_r \colon \Sigma_r \to \Sigma$ and a sentence $r \in \mathrm{Sen}^{\mathcal{I}}(\Sigma_r)$.

  For every signature morphism $\varphi \colon \Sigma \to \Sigma'$, the translation of a Σ-sentence $\langle i_r, r \rangle$ via φ is $\mathrm{Sen}^{\mathrm{INT}}(\varphi)(i_r, r) = \langle i'_r \colon \Sigma'_r \to \Sigma', \mathrm{Sen}^{\mathcal{I}}(\varphi_r)(r) \rangle$, where i'_r and φ_r are the monic and epi of the unique factorization of $(i_r\,;\varphi)$:

$$\begin{tikzcd}
\Sigma_r \arrow[r, "\varphi_r"] \arrow[d, "i_r"'] & \Sigma'_r \arrow[d, "i'_r"] \\
\Sigma \arrow[r, "\varphi"'] & \Sigma'
\end{tikzcd}$$

- For every Σ-model M and Σ-sentence $\langle i_r, r \rangle$, $M \models^{\mathrm{INT}}_{\Sigma} \langle i_r, r \rangle = M \models^{\mathcal{I}}_{\Sigma} i_r(r)$.

PROPOSITION 5.1.2. *Given an $\mathbb{RL}$-institution $\mathcal{I}$ for which the category of signatures is equipped with a factorization system such that any signature morphism has a unique factorization, the construction $\mathrm{INT}(\mathcal{I})$ is an $\mathbb{RL}$-institution.*

PROOF. The functoriality of $\mathrm{Sen}^{\mathrm{INT}}$ is ensured by the functoriality of $\mathrm{Sen}^{\mathcal{I}}$ and the uniqueness of the factorization of any signature morphism: for any two composable signature morphisms $\varphi \colon \Sigma \to \Sigma'$, $\varphi' \colon \Sigma' \to \Sigma''$,

$$\mathrm{Sen}^{\mathrm{INT}}(\varphi\,;\varphi')(i_r, r) = \mathrm{Sen}^{\mathrm{INT}}(\varphi')(\mathrm{Sen}^{\mathrm{INT}}(\varphi)(i_r, r)).$$

$$\begin{tikzcd}
\Sigma_r \arrow[r, "\varphi_r"] \arrow[d, "i_r"'] & \Sigma'_r \arrow[r, "\varphi'_r"] \arrow[d, "i'_r"] & \Sigma''_r \arrow[d, "i''_r"] \\
\Sigma \arrow[r, "\varphi"'] & \Sigma' \arrow[r, "\phi'"'] & \Sigma''
\end{tikzcd}$$

[1] This is reminiscent of the notion of inclusion system from [DGS93].

[2] With respect to the factorization system of $\mathbb{S}\mathrm{ig}^{\mathcal{I}}$.

The satisfaction condition follows from

$$\begin{aligned} M' \models^{\mathrm{INT}}_{\Sigma'} \varphi(i_r, r) \quad & \text{iff} \quad M' \models^{\mathrm{INT}}_{\Sigma'} \langle i'_r, \varphi_r(r) \rangle \\ & \text{iff} \quad M' \models^{\mathcal{J}}_{\Sigma'} i'_r(\varphi_r(r)) \\ & \text{iff} \quad M' \models^{\mathcal{J}}_{\Sigma'} \varphi(i_r(r)) \\ & \text{iff} \quad M' {\restriction_{\varphi}} \models^{\mathcal{J}}_{\Sigma} i_r(r) \\ & \text{iff} \quad M' {\restriction_{\varphi}} \models^{\mathrm{INT}}_{\Sigma} \langle i_r, r \rangle . \end{aligned}$$

□

FACT 5.1.3. The logics CSP($\mathcal{FOL}$) and $\mathcal{BGP}$ satisfy the conditions in the definition of INT($\mathcal{J}$).

LOGIC PROGRAMS

We define a many-valued logical framework for writing programs based on an arbitrary $\mathbb{RL}$-institution whose sentences are regarded in what follows as *atomic*. Since we are interested in expressing clauses and queries, which are in essence universally and existentially quantified sentences, the base $\mathbb{RL}$-institution must permit the addition of quantifiers to its sentences via signature extensions with sets of variables. To ensure that the variables thus added are translated properly along signature morphisms, we take into account only those extensions of signatures that belong to a *quantification space* – a notion originating from [Dia10] in the context of quasi-Boolean encodings, that has been used in several studies on many-valued institutions [Dia13] and institution-independent logic programming [ŢF15a]. Despite such recent developments, it is worth noting that the main idea behind quantification spaces has been previously presented in the literature under various forms, starting with [Tar86], which is one of the first works that considered open formulae in arbitrary institutions. We adapt the definition of a quantification space given in [ŢF15a] to the many-valued setting, starting from the following result concerning arrow categories:

FACT 5.1.4. Given a category $\mathbb{K}$ and a subcategory $\mathbb{Q}$ of the arrow category $\mathbb{K}^{\rightarrow}$, the domain functor $\mathrm{dom}\colon \mathbb{Q} \to \mathbb{K}$ gives rise to a natural transformation $\iota_{\mathbb{Q}}\colon (\_ \,/\, \mathbb{Q}) \Rightarrow \mathrm{dom}^{\mathrm{op}} ; (\_ \,/\, \mathbb{K})$ where, for every object $\langle A_1, f, A_2 \rangle \in |\mathbb{Q}|$ (i.e. arrow in $\mathbb{K}$), $\iota_{\mathbb{Q},f}\colon f \,/\, \mathbb{Q} \Rightarrow A_1 \,/\, \mathbb{K}$ is the functor that maps the morphisms $\langle g_1, g_2 \rangle\colon \langle A_1, f, A_2 \rangle \to \langle A'_1, f', A'_2 \rangle$ in $\mathbb{Q}$ (corresponding to commutative squares in $\mathbb{K}$) to $g_1\colon A_1 \to A'_1$:

$$\begin{tikzcd} A_1 \arrow[r, "g_1"] \arrow[d, "f"'] & A_1' \arrow[d, "f'"] \\ A_2 \arrow[r, "g_2"'] & A_2' \end{tikzcd} \quad \mapsto \quad \begin{tikzcd} A_1 \arrow[r, "g_1"] & A_1' \end{tikzcd}$$

DEFINITION 5.1.5 (Quantification space). A *quantification space* for an $\mathbb{RL}$-institution $\mathcal{I}\colon \mathbb{S}\mathrm{ig} \to \mathbb{RL}\text{-}\mathbb{R}\mathrm{oom}$ consists of a subcategory $\mathbb{Q}$ of $\mathbb{S}\mathrm{ig}^{\to}$ such that:

1. every arrow in $\mathbb{Q}$ corresponds to a pushout in $\mathbb{S}\mathrm{ig}$, and
2. the natural transformation $\iota_{\mathbb{Q}}\colon (\_\,/\,\mathbb{Q}) \Rightarrow \mathrm{dom}^{\mathrm{op}}\,;(\_\,/\,\mathbb{S}\mathrm{ig})$ is an isomorphism.

This means that for every signature extension $\chi\colon \Sigma \to \Sigma(\chi)$ in $|\mathbb{Q}|$ and every signature morphism $\varphi\colon \Sigma \to \Sigma'$ there exist a unique extension $\chi'\colon \Sigma' \to \Sigma'(\chi')$ in $|\mathbb{Q}|$ and a unique signature morphism $\phi\colon \Sigma(\chi) \to \Sigma'(\chi')$ such that the pair $\langle \varphi, \phi\rangle$ defines a morphism in $\mathbb{Q}$ between the arrows χ and χ' (moreover, χ' and ϕ correspond to a pushout of φ and χ).

We denote hereafter the signature extension χ' by $\chi^{\varphi}\colon \Sigma' \to \Sigma'(\chi^{\varphi})$ and the signature morphism ϕ by $\varphi^{\chi}\colon \Sigma(\chi) \to \Sigma'(\chi^{\varphi})$.

$$\begin{tikzcd} \Sigma \arrow[r, "\varphi"] \arrow[d, "\chi"'] & \Sigma' \arrow[d, "\chi^{\varphi}"] \\ \Sigma(\chi) \arrow[r, "\varphi^{\chi}"'] & \Sigma'(\chi^{\varphi}) \end{tikzcd}$$

EXAMPLE 5.1.6.

1. Quantification space for $\mathcal{PL}_{\mathbb{RL}}$: For any many-valued institution, and in particular for the institution of many-valued propositional logic, we can define a trivial quantification space whose signature extensions are all identity morphisms (there are no variables in that case).
2. Quantification space for $\mathcal{FOL}_{\mathbb{RL}}$: The standard quantification space for first-order logic extends to many-valued first-order logic by considering signature extensions of the form $\chi\colon \langle \mathcal{L}, S, F, P\rangle \to \langle \mathcal{L}, S, F + X, P\rangle$, where X is a set of variables. For any signature morphism $\langle \ell, \varphi\rangle\colon \langle \mathcal{L}, S, F, P\rangle \to \langle \mathcal{L}', S', F', P'\rangle$, there exist:

- a unique signature extension $\chi^{\langle \ell, \varphi\rangle}\colon \langle \mathcal{L}', S', F', P'\rangle \to \langle \mathcal{L}', S', F' + X^{\varphi}, P'\rangle$, with $X^{\varphi} = \{x : \varphi^{\mathrm{st}}(s) \mid x : s \in X\}$ as in the definition of the first-order institution $\mathcal{FOL}$, and

- a unique morphism $\langle \ell, \varphi \rangle^{\chi} \colon \langle \mathcal{L}, S, F \uplus X, P \rangle \to \langle \mathcal{L}', S', F' \uplus X^{\varphi}, P' \rangle$, which is the canonical extension of $\langle \ell, \varphi \rangle$ that maps, as in the definition of the first-order institution $\mathcal{FOL}$, every variable $x : s$ for $\langle \mathcal{L}, S, F, P \rangle$ to $x : \varphi^{\mathrm{st}}(s)$. These variables are potentially annotated with the signatures over which they are defined – as in $(x : s, \langle \mathcal{L}, S, F, P \rangle)$ – in order to avoid clashes with other symbols from those signatures.

3. Quantification space for CSP($\mathcal{FOL}$): We define a quantification space similarly to the previous example; we consider signature morphisms $\langle \ell_{\chi}, \chi \rangle \colon \langle \mathcal{L}, \Sigma, V, E \rangle \to \langle \mathcal{L}_{\chi}, \Sigma_{\chi}, V, E_{\chi} \rangle$ that add arbitrary new operation symbols to the first-order signature Σ and allow the change of the underlying lattice. In other words, we consider pairs $\langle \ell_{\chi}, \chi \rangle$ of monic residuated lattice morphisms $\ell_{\chi} \colon \mathcal{L}' \to \mathcal{L}$ and presentation morphisms $\chi \colon \langle \Sigma, E \rangle \to \langle \Sigma_{\chi}, E_{\chi} \rangle$, where $\Sigma = \langle S, F, P \rangle$, $\Sigma_{\chi} = \langle S, F + X, P \rangle$, and $E_{\chi} = \chi(E)$, with every new operation symbol from X of the form $x \colon w \to s$, or more explicitly $(x \colon w \to s, \Sigma)$. Then, for any signature morphism $\langle \ell_{\varphi}, \varphi \rangle \colon \langle \mathcal{L}, \Sigma, V, E \rangle \to \langle \mathcal{L}', \Sigma', V', E' \rangle$, there exist:

- a unique extension $\langle \ell_{\chi^{\varphi}}, \chi^{\varphi} \rangle \colon \langle \mathcal{L}', \Sigma', V', E' \rangle \to \langle \mathcal{L}'_{\chi^{\varphi}}, \Sigma'_{\chi^{\varphi}}, V', E'_{\chi^{\varphi}} \rangle$, where $\mathcal{L}'_{\chi^{\varphi}}$ is the vertex of the pullback of ℓ_{χ} and ℓ_{φ}, and $\Sigma'_{\chi^{\varphi}} = \langle S', F' + X^{\varphi}, P' \rangle$ is the extension of Σ' with operation symbols of the form $(x \colon \varphi^{\mathrm{st}}(w) \to \varphi^{\mathrm{st}}(s), \Sigma')$ for symbols $(x \colon w \to s, \Sigma)$ in Σ_{χ}, and
- a unique morphism $\langle \ell_{\varphi^{\chi}}, \varphi^{\chi} \rangle \colon \langle \mathcal{L}_{\chi}, \Sigma_{\chi}, V, E_{\chi} \rangle \to \langle \mathcal{L}'_{\chi^{\varphi}}, \Sigma'_{\chi^{\varphi}}, V', E'_{\chi^{\varphi}} \rangle$ that extends φ by mapping the symbols of the form (x, Σ) to (x, Σ').

4. Quantification space for $\mathcal{BGP}$: In a similar way, we can define a quantification space for the $\mathbb{RL}$-institution $\mathcal{BGP}$ by considering extensions of signatures with new formation symbols, instead of operation symbols.

REMARK 5.1.7. Note that since INT($\mathcal{I}$) has the same signatures as $\mathcal{I}$, any quantification space for $\mathcal{I}$ is also a valid quantification space for INT($\mathcal{I}$).

DEFINITION 5.1.8 (Adequacy). For any $\mathbb{RL}$-institution, a quantification space $\mathbb{Q}$ is *adequate* if every arrow $\langle \varphi, \varphi^{\chi} \rangle \colon \chi \to \chi^{\varphi}$ in $\mathbb{Q}$ corresponds to a model-amalgamation square: for every Σ'-model M' and $\Sigma(\chi)$-model N such that $M' \restriction_{\varphi} = N \restriction_{\chi}$ there exists a unique model N' of $\Sigma'(\chi^{\varphi})$ – the *amalgamation* of M' and N – such that $N' \restriction_{\chi^{\varphi}} = M'$ and $N' \restriction_{\varphi^{\chi}} = N$.

REMARK 5.1.9. All quantification spaces presented in Example 5.1.6 are adequate for their corresponding institutions.

REMARK 5.1.10. Because the morphisms of any quantification space $\mathbb{Q}$ form a category, for every signature extension $\chi \colon \Sigma \to \Sigma(\chi)$ in $|\mathbb{Q}|$ and every pair of composable signature morphisms $\varphi \colon \Sigma \to \Sigma'$ and $\varphi' \colon \Sigma' \to \Sigma''$, we have $(\chi^{\varphi})^{\varphi'} = \chi^{\varphi;\varphi'}$ and $\varphi^{\chi}\,;(\varphi')^{\chi^{\varphi}} = (\varphi\,;\varphi')^{\chi}$. Moreover, $\chi^{\mathrm{id}_{\Sigma}} = \chi$ and ${\mathrm{id}_{\Sigma}}^{\chi} = \mathrm{id}_{\Sigma(\chi)}$. Quantification spaces provide therefore a mechanism for translating signature extensions along morphisms of signatures in a functorial manner, by means of dedicated pushout constructions.

$$\begin{tikzcd}
\Sigma \arrow[r, "\varphi"] \arrow[d, "\chi"'] & \Sigma' \arrow[r, "\varphi'"] \arrow[d, "\chi^{\varphi}"] & \Sigma'' \arrow[d, "(\chi^{\varphi})^{\varphi'} = \chi^{\varphi;\varphi'}"] \\
\Sigma(\chi) \arrow[r, "\varphi^{\chi}"'] \arrow[rr, bend right, "(\varphi\,;\varphi')^{\chi}"'] & \Sigma'(\chi^{\varphi}) \arrow[r, "(\varphi')^{\chi^{\varphi}}"'] & \Sigma''((\chi^{\varphi})^{\varphi'})
\end{tikzcd}$$

DEFINITION 5.1.11 (Conservative signature extension). A signature extension $\chi\colon \Sigma \to \Sigma(\chi)$ in $|\mathbb{Q}|$ is *conservative* if the model-reduct functor $\mathrm{Mod}^{\mathcal{I}}(\chi)\colon \mathrm{Mod}^{\mathcal{I}}(\Sigma(\chi)) \to \mathrm{Mod}^{\mathcal{I}}(\Sigma)$ is surjective on objects.

EXAMPLE 5.1.12. In the case of relational first-order logic, a signature extension is conservative if the sets of symbols of constants are not empty, for any sort.

THE $\mathbb{RL}$-INSTITUTION OF MANY-VALUED $\mathcal{I}$-LOGIC PROGRAMS.

In what follows, we show how the traditional construction of first-order logic over an arbitrary 'atomic' institution can be generalized to $\mathbb{RL}$-institutions to provide support for many-valued logic programs.

DEFINITION 5.1.13. For any $\mathbb{RL}$-institution $\mathcal{I}$ with an adequate quantification space $\mathbb{Q}$, we define $\mathrm{LP}(\mathcal{I},\mathbb{Q})$ as the tuple $\langle \mathbb{S}\mathrm{ig}^{\mathrm{LP}}, \mathrm{Sen}^{\mathrm{LP}}, \mathrm{Mod}^{\mathrm{LP}}, \mathrm{RL}^{\mathrm{LP}}, \models^{\mathrm{LP}}\rangle$ where:

- $\mathbb{S}\mathrm{ig}^{\mathrm{LP}} = \mathbb{S}\mathrm{ig}^{\mathcal{I}}$,
- $\mathrm{Mod}^{\mathrm{LP}} = \mathrm{Mod}^{\mathcal{I}}$,
- $\mathrm{RL}^{\mathrm{LP}} = \mathrm{RL}^{\mathcal{I}}$,
- For every signature Σ, $\mathrm{Sen}^{\mathrm{LP}}(\Sigma)$ is the smallest set that contains the sentences from $\mathrm{Sen}^{\mathcal{I}}(\Sigma)$ as atoms, and is closed under Boolean connectives and first-order quantifiers. From these, we distinguish two types of sentences:
  - *clauses*: $\forall X.\rho$,
    where $X\colon \Sigma \to \Sigma(X)$ is a conservative extension, and ρ is of the form $\overline{H} \to C$, where $\overline{H} = H_1 \wedge H_2 \wedge \cdots \wedge H_n$ and $H_1, H_2, \ldots, H_n, C$ are atomic sentences – that is, sentences from $\mathrm{Sen}^{\mathcal{I}}(\Sigma(X))$.
  - *queries*: $\exists X.\rho$,
    where $X\colon \Sigma \to \Sigma(X)$ is an extension, and ρ is of the form $Q_1 \wedge Q_2 \wedge \cdots \wedge Q_m$, with $Q_1, Q_2, \ldots, Q_m$ sentences from $\mathrm{Sen}^{\mathcal{I}}(\Sigma(X))$.

For every signature morphism $\varphi\colon \Sigma \to \Sigma'$, the sentence translation is defined inductively, as for the institution of first-order logic, starting with the translation of atoms $\rho \in \mathrm{Sen}^{\mathcal{I}}(\Sigma)$ defined as in $\mathcal{I}$: $\mathrm{Sen}^{\mathrm{LP}}(\varphi)(\rho) = \mathrm{Sen}^{\mathcal{I}}(\varphi)(\rho)$.

- The satisfaction of sentences by models is defined once again inductively, starting from the satisfaction of atoms as in $\mathcal{I}$: for every Σ-model M, and every $\rho \in \mathrm{Sen}^{\mathcal{I}}(\Sigma)$, $(M \vDash^{\mathrm{LP}}_{\Sigma} \rho) = (M \vDash^{\mathcal{I}}_{\Sigma} \rho)$.

PROPOSITION 5.1.14. *Given an $\mathbb{RL}$-institution $\mathcal{I}$ equipped with an adequate quantification space $\mathbb{Q}$, $\mathrm{LP}(\mathcal{I}, \mathbb{Q})$ is an $\mathbb{RL}$-institution.*

PROOF. The functoriality of $\mathrm{Sen}^{\mathrm{LP}}$ is an immediate consequence of the functoriality of $\mathrm{Sen}^{\mathcal{I}}$, while the satisfaction condition can be proved inductively on the structure of sentences. We focus on the case of universally quantified sentences; the result for existential sentences can be proved similarly. Let us consider an arbitrary signature morphism $\varphi\colon \Sigma \to \Sigma'$, a Σ'-model M' and a Σ-sentence $\forall X.\rho$. Then,

$$(M'\restriction_{\varphi} \vDash^{\mathrm{LP}}_{\Sigma} \forall X.\rho) = \mathrm{RL}^{\mathrm{LP}}(X)\big(\bigwedge\{N \vDash^{\mathrm{LP}}_{\Sigma(X)} \rho \mid N\restriction_X = M'\restriction_{\varphi}\}\big) \tag{5.1}$$

From the adequacy of the quantification space $\mathbb{Q}$, we have that for any $\Sigma(X)$-model N such that $N\restriction_X = M'\restriction_{\varphi}$ there exists a unique model N' of $\Sigma'(X^{\varphi})$ such that $N'\restriction_{\varphi^X} = N$ and $N'\restriction_{X^{\varphi}} = M'$. We use $N'\restriction_{\varphi^X} = N$, and (5.1) becomes

$$\mathrm{RL}^{\mathrm{LP}}(X)\big(\bigwedge\{N'\restriction_{\varphi^X} \vDash^{\mathrm{LP}}_{\Sigma(X)} \rho \mid (N'\restriction_{\varphi^X})\restriction_X = M'\restriction_{\varphi}\}\big). \tag{5.2}$$

From the satisfaction condition for the signature morphism φ^X and sentences without quantifiers, we have that for any $\Sigma'(X^{\varphi})$-model N', and any $\Sigma(X)$-sentence ρ,

$$\mathrm{RL}^{\mathrm{LP}}(X^{\varphi})(N' \vDash^{\mathrm{LP}}_{\Sigma'(X^{\varphi})} \varphi^X(\rho)) = (N'\restriction_{\varphi^X} \vDash^{\mathrm{LP}}_{\Sigma(X)} \rho).$$

(5.2) becomes now

$$\mathrm{RL}^{\mathrm{LP}}(X)\big(\bigwedge\{\mathrm{RL}^{\mathrm{LP}}(\varphi^X)\big(N' \vDash^{\mathrm{LP}}_{\Sigma'(X^{\varphi})} \varphi^X(\rho)\big) \mid N'\restriction_{X^{\varphi}} = M'\}\big) \tag{5.3}$$

Because $\mathrm{RL}^{\mathrm{LP}}(\varphi^X) \mathbin{;} \mathrm{RL}^{\mathrm{LP}}(X) = \mathrm{RL}^{\mathrm{LP}}(X^{\varphi}) \mathbin{;} \mathrm{RL}^{\mathrm{LP}}(\varphi)$, (5.3) equals

$$\begin{aligned}
&\mathrm{RL}^{\mathrm{LP}}(\varphi)\big(\mathrm{RL}^{\mathrm{LP}}(X^{\varphi})\big(\bigwedge\{N' \vDash^{\mathrm{LP}}_{\Sigma'(X^{\varphi})} \varphi^X(\rho) \mid N'\restriction_{X^{\varphi}} = M'\}\big)\big)\\
&= \mathrm{RL}^{\mathrm{LP}}(\varphi)\big(M' \vDash^{\mathrm{LP}}_{\Sigma'} \forall X^{\varphi}.\varphi^X(\rho)\big)\\
&= \mathrm{RL}^{\mathrm{LP}}(\varphi)(M' \vDash^{\mathrm{LP}}_{\Sigma'} \varphi(\forall X.\rho)).
\end{aligned}$$

□

For the rest of this section, we fix an arbitrary $\mathbb{RL}$-institution $\mathrm{LP}(\mathcal{I}, \mathbb{Q})$; often, it is useful to denote it just by LP, and to consider it a functor from $\mathbb{S}\mathrm{ig}^{\mathrm{LP}}$ into $\mathbb{RL}$-$\mathbb{R}\mathrm{oom}$. For simplicity, we often omit the superscripts when there is no risk of confusion.

DEFINITION 5.1.15 (Logic program). A *many-valued logic program* over a signature Σ is a presentation $\langle \Sigma, E \rangle$ together with a set Γ of clauses over Σ.

In classical logic programming, the sets of sentences E and Γ coincide. Distinguishing between the two sets allows us to consider different degrees of confidence for the clauses $\forall X.\rho$ of Γ.

DEFINITION 5.1.16 (η-correctness of a clause). Given a many-valued logic program $\langle \Sigma, E, \Gamma \rangle$, a clause $\forall X.\rho$ is said to be *η-correct* if $(E \vDash_{\Sigma} \forall X.\rho) \geq \eta$.

Obviously, in the classical setting, all clauses are 1-correct, as they belong to the presentation $\langle \Sigma, E \rangle$. In the context of service-oriented computing, E is empty, which means that the correctness of a clause $\forall X.\overline{H} \to C$ is given directly by the value with which C is a consequence of $\overline{H}$, i.e. $\overline{H} \vDash_{\Sigma(X)} C$.

DEFINITION 5.1.17 (Logic-programming problem). A logic-programming problem for a signature Σ is given by a many-valued logic program $\langle \Sigma, E, \Gamma \rangle$ and by a query $\exists X.\rho$ over Σ. Furthermore, an instance of a many-valued logic-programming problem is a tuple $\langle \Sigma, E, \Gamma, \exists X.\rho, \mu \rangle$, where the new component, μ, is a required confidence value from $\mathrm{RL}(\Sigma)$.

In other words, given a logic program $\langle \Sigma, E, \Gamma \rangle$, we are interested in finding a solution to a query $\exists X.\rho$ that is guaranteed to be appropriate with at least the value μ. Solutions are a particular kind of substitutions between signature extensions.

DEFINITION 5.1.18 (Substitution). Given two signature extensions $X \colon \Sigma \to \Sigma(X)$ and $Y \colon \Sigma \to \Sigma(Y)$ from $|\mathbb{Q}|$, a *substitution* $\psi \colon X \to Y$ is a corridor $\psi \colon \mathrm{LP}(\Sigma(X)) \to \mathrm{LP}(\Sigma(Y))$ from $\mathbb{RL}$-$\mathbb{R}\mathrm{oom}$ such that the diagram below commutes:

$$\begin{tikzcd}
 & \mathrm{LP}(\Sigma) \arrow[dl, "\mathrm{LP}(X)"'] \arrow[dr, "\mathrm{LP}(Y)"] & \\
\mathrm{LP}(\Sigma(X)) \arrow[rr, "\psi"'] & & \mathrm{LP}(\Sigma(Y))
\end{tikzcd}$$

The substitution ψ induces:

- a translation of sentences, denoted by $\psi(\rho)$ for every $\Sigma(X)$-sentence ρ,
- a reduction of models, denoted by $N\restriction_{\psi}$ for every $\Sigma(Y)$-model N, and
- a morphism of residuated lattices, denoted by $\mathrm{RL}(\psi)$

such that a satisfaction condition like the one for many-valued corridors from Definition 3.1.6 holds.

The nature of the concept of substitution that we have presented above is semantic; this is a consequence of the direct translation of sentences, reduction of models and change of truth spaces. In practice, substitutions are defined through syntactical norms. For example, in classical first-order logic programming, substitutions are functions from sets of variables to terms: every variable is mapped to a term of corresponding sort. In the context of service-oriented computing, for institutions like INT(CSP($\mathcal{I}$)) and INT($\mathcal{BGP}$), substitutions from $X\colon \Sigma \to \Sigma(X)$ to $Y\colon \Sigma \to \Sigma(Y)$ are defined as morphisms of signatures $\psi\colon \Sigma(X) \to \Sigma(Y)$ that make the following diagram commute:

$$\begin{tikzcd}
 & \Sigma \arrow[dl, "X"'] \arrow[dr, "Y"] & \\
\Sigma(X) \arrow[rr, "\psi"'] & & \Sigma(Y)
\end{tikzcd}$$

REMARK 5.1.19. Working with substitutions and signature extensions implies dealing with values from multiple lattices. For the remaining part of this chapter, to simplify notations, we automatically map all these values (through the underlying lattice morphisms of signature extensions or of substitutions) to the underlying lattice of the base signature.

DEFINITION 5.1.20 (Solution to a problem). A substitution $\psi\colon X \to Y$ is a *ζ-solution to a problem* $\langle \Sigma, E, \Gamma, \exists X.\rho \rangle$ if Y is conservative and $(E \models_{\Sigma} \forall Y.\psi(\rho)) \geq \zeta$.

DEFINITION 5.1.21 (Solution to an instance of a problem). A substitution $\psi\colon X \to Y$ is a *solution* to an instance of a logic-programming problem $\langle \Sigma, E, \Gamma, \exists X.\rho, \mu \rangle$ if it is a μ-solution to $\langle \Sigma, E, \Gamma, \exists X.\rho \rangle$.

The next result corresponds to one of Herbrand's theorems for classical logical programming, which states that, if there exists a solution to a query, then the query is satisfiable – in a Boolean sense [Llo87].

PROPOSITION 5.1.22 (Soundness). *If $\psi\colon X \to Y$ is a ζ-solution to a problem $\langle \Sigma, E, \Gamma, \exists X.\rho \rangle$, then $(E \models_{\Sigma} \exists X.\rho) \geq \zeta$.*

PROOF. We start from

$$(E \models_{\Sigma} \exists X.\rho) = \bigwedge\{(M \models_{\Sigma} E) \to (M \models_{\Sigma} \exists X.\rho) \mid M \in |\mathrm{Mod}(\Sigma)|\}.$$

It thus suffices to show that, for any model $M \in |\mathrm{Mod}(\Sigma)|$,

$$\big((M \models_{\Sigma} E) \to (M \models_{\Sigma} \exists X.\rho)\big) \geq \zeta,$$

which, by the adjunction between multiplication and residuum, is equivalent to

$$(M \models_{\Sigma} \exists X.\rho) \geq (M \models_{\Sigma} E) * \zeta. \tag{5.4}$$

Because ψ is a ζ-solution, we know that $\big(E \models_{\Sigma} \forall Y.\psi(\rho)\big) \geq \zeta$, so

$$\big((M \models_{\Sigma} E) \to (M \models_{\Sigma} \forall Y.\psi(\rho))\big) \geq \zeta,$$

which is equivalent to

$$(M \models_{\Sigma} \forall Y.\psi(\rho)) \geq \big((M \models_{\Sigma} E) * \zeta\big). \tag{5.5}$$

From (5.4) and (5.5) it follows that it suffices to compare $M \models_{\Sigma} \forall Y.\psi(\rho)$ and $M \models_{\Sigma} \exists X.\rho$. To that end, we have that $M \models_{\Sigma} \forall Y.\psi(\rho)$ is equal to

$$\bigwedge\{\mathrm{RL}(Y)(M_Y \models_{\Sigma(Y)} \psi(\rho)) \mid M_Y \in |\mathrm{Mod}(\Sigma(Y))|, M_Y\!\restriction_Y = M\},$$

which, because $\mathrm{RL}(Y) = \mathrm{RL}(\psi) \,;\mathrm{RL}(X)$, is equal to

$$\bigwedge\{\mathrm{RL}(X)\big(\mathrm{RL}(\psi)(M_Y \models_{\Sigma(Y)} \psi(\rho))\big) \mid M_Y \in |\mathrm{Mod}(\Sigma(Y))|, M_Y\!\restriction_Y = M\}. \tag{5.6}$$

From the satisfaction condition for ψ, we have that $\mathrm{RL}(\psi)(M_Y \models_{\Sigma(Y)} \psi(\rho)) = (M_Y\!\restriction_{\psi} \models_{\Sigma(X)} \rho)$, and thus (5.6) is equal to

$$\begin{gathered}
\bigwedge\{\mathrm{RL}(X)(M_Y\!\restriction_{\psi} \models_{\Sigma(X)} \rho) \mid M_Y \in |\mathrm{Mod}(\Sigma(Y))|, (M_Y\!\restriction_{\psi})\!\restriction_X = M\} \\
\leq \\
\bigvee\{\mathrm{RL}(X)(M_X \models_{\Sigma(X)} \rho) \mid M_X \in |\mathrm{Mod}(\Sigma(X))|, (M_X\!\restriction_X) = M\}
\end{gathered}$$

which corresponds to the definition of $M \models_{\Sigma} \exists X.\rho$. □

5.2 OPERATIONAL SEMANTICS

As in conventional logic programmig, solutions to logic-programming problems in the sense of Definition 5.1.17 (or to problem instances) can be computed by means of unification and resolution. For that purpose, we need to adapt these two key concepts to our many-valued setting, which we do by extending the already general notions from [ŢF15b] to $\mathbb{RL}$-institutions.

DEFINITION 5.2.1 (Unifier). A *unifier* for two atomic sentences $\rho_1 \in \mathrm{Sen}(X_1)$, $\rho_2 \in \mathrm{Sen}(X_2)$, is any pair of substitutions $\psi_1 \colon X_1 \to Y$ and $\psi_2 \colon X_2 \to Y$.

In practice, the sentence ρ_1 corresponds to an atomic part of a query, while ρ_2 corresponds to the conclusion of a clause. Notice that we do not impose the usual restriction $\psi_1(\rho_1) = \psi_2(\rho_2)$ on the substitutions; instead, we are interested in the value of the entailment $\psi_2(\rho_2) \vDash_{\Sigma(Y)} \psi_1(\rho_1)$. This makes the present notion of unifier fundamentally different from the classical one, as the entailment is not symmetric.

The reason why we adopt it is that this new formulation is suitable for modelling the service matching presented in Chapter 4: a unifier is used in the context of service-discovery processes, where the role of the substitution ψ_1 is played by the attachment morphisms ϕ. As expected, we can compare unifiers through the entailment value $\psi_2(\rho_2) \vDash_{\Sigma(Y)} \psi_1(\rho_1)$, which should be seen as a way to evaluate their suitability for the task. We thus obtain the notion of a *maximal unifier* – once again, not compatible with the classical notion of most general unifier. The choice of such a maximal unifier captures the service selection process presented previously; there the process chooses a maximal unifier with an identity for the second component.

COMPUTING SOLUTIONS TO PROBLEMS

DEFINITION 5.2.2 (Resolution for problems). Let $\langle \Sigma, E, \Gamma, \exists X_1.\alpha \wedge Q_i \wedge \omega \rangle$ be a logic-programming problem with α and ω conjunctions of atoms, and $\forall X_2.\overline{H} \to C$ a Σ-clause from Γ. A Σ-query $\exists Y.\psi_1(\alpha) \wedge \psi_2(\overline{H}) \wedge \psi_1(\omega)$ is said to be *derived by resolution* with a degree of confidence δ from $\exists X_1.\alpha \wedge Q_i \wedge \omega$ and $\forall X_2.\overline{H} \to C$ using the unifier $\langle \psi_1 \colon X_1 \to Y, \psi_2 \colon X_2 \to Y \rangle$ if $(\psi_2(C) \vDash_{\Sigma(Y)} \psi_1(Q_i)) \geq \delta$.

$$\frac{\exists X_1 \cdot \alpha \wedge Q_i \wedge \omega \qquad\qquad \forall X_2 \cdot C \leftarrow \overline{H}}{\exists Y \cdot \psi_1(\alpha) \wedge \psi_2(\overline{H}) \wedge \psi_1(\omega)} \; \langle \psi_1, \psi_2 \rangle$$

The following meta-rules of generalization and substitutivity are needed to prove the soundness of resolution, in the sense that it preserves solutions.

PROPOSITION 5.2.3 (Generalization rule). *Given a conservative signature extension $X\colon \Sigma \longrightarrow \Sigma(X)$, for any set E of Σ-sentences and any $\Sigma(X)$-sentence ρ, $(E \vDash_{\Sigma(X)} \rho) = (E \vDash_{\Sigma} \forall X.\rho)$.*

PROOF. We recall that

$$E \vDash_{\Sigma(X)} \rho = \bigwedge_{M_x \in |\mathrm{Mod}(\Sigma(X))|} (M_x \vDash_{\Sigma(X)} E) \longrightarrow (M_x \vDash_{\Sigma(X)} \rho).$$

On the other hand, since X is conservative,

$$\begin{aligned} E \vDash_{\Sigma} \forall X.\rho &= \bigwedge_{M \in |\mathrm{Mod}(\Sigma)|} (M \vDash_{\Sigma} E) \longrightarrow (M \vDash_{\Sigma} \forall X.\rho) \\ &= \bigwedge_{M \in |\mathrm{Mod}(\Sigma)|} (M \vDash_{\Sigma} E) \longrightarrow \bigwedge_{M_x \restriction_X = M} (M_x \vDash_{\Sigma(X)} \rho) \\ &= \bigwedge_{M_x \in |\mathrm{Mod}(\Sigma(X))|} (M_x \vDash_{\Sigma(X)} E) \longrightarrow (M_x \vDash_{\Sigma(X)} \rho). \end{aligned}$$

□

PROPOSITION 5.2.4 (Substitutivity rule). *For any set E of Σ-sentences, any $\Sigma(X)$-sentence ρ, and any substitution $\psi\colon X \longrightarrow Y$,*

$$(E \vDash_{\Sigma(X)} \rho) \leq (E \vDash_{\Sigma(Y)} \psi(\rho)).$$

PROOF. By the definition of graded consequence,

$$E \vDash_{\Sigma(Y)} \psi(\rho) = \bigwedge_{M_Y \in |\mathrm{Mod}(\Sigma(Y))|} (M_Y \vDash_{\Sigma(Y)} E) \longrightarrow (M_Y \vDash_{\Sigma(Y)} \psi(\rho)). \tag{5.7}$$

From the satisfaction condition for substitution ψ, $(M_Y \vDash_{\Sigma(Y)} \psi(\rho)) = (M_Y \restriction_{\psi} \vDash \rho)$ for any model $M_Y \in |\mathrm{Mod}(\Sigma(Y))|$, and thus (5.7) equals

$$\bigwedge_{M_X \in \mathrm{Mod}(\Sigma(X))} (M_X \vDash_{\Sigma(X)} E) \longrightarrow \bigwedge_{\substack{M_Y \in |\mathrm{Mod}(\Sigma(Y))| \\ M_Y \restriction_{\psi} = M_X}} M_Y \restriction_{\psi} \vDash_{\Sigma(X)} \rho$$

$$\geq$$

$$\bigwedge_{M_X \in \mathrm{Mod}(\Sigma(X))} (M_X \vDash_{\Sigma(X)} E) \longrightarrow \bigwedge_{M_X \in |\mathrm{Mod}(\Sigma(X))|} M_X \vDash_{\Sigma(X)} \rho.$$

□

DEFINITION 5.2.5 (Modus ponens). For two Σ-sentences p and q, we denote by $\mathrm{mp}(p,q)$ the value with which $\{p, p \to q\}$ entails q, i.e. $\{p, p \to q\} \vDash_\Sigma q$.

REMARK 5.2.6. For any signature Σ having a Heyting algebra as its underlying lattice, $\mathrm{mp}(p,q) = 1$ for any Σ-sentences p and q; this means that the value of modus ponens is independent of p and q. However, for any signature Σ having a non-Heyting underlying lattice, such as the real-valued Łukasiewicz lattice $\mathcal{L}_{[0,1]}$ from Example 2.3.8, $\mathrm{mp}(p,q) \leq 1$, possibly strictly, and depends on the sentences p and q. For example, for a signature Σ of the many-valued first-order logic programming institution $\mathrm{LP}(\mathcal{FOL}_{\mathbb{RL}})$ with $\mathrm{RL}(\Sigma) = \mathcal{L}_3$ and containing two nullary predicates p and q, we have that $\mathrm{mp}(p,q) = 1$ when the interpretations of p and q coincide, and $\mathrm{mp}(p,q) = 0.5$ when the interpretations of p and q differ.

To simplify things, we denote by $\mathrm{mp}_{\mathcal{L}}$, or simply by mp, the smallest modus-ponens value that can be obtained for a lattice $\mathcal{L}$: $\mathrm{mp}_{\mathrm{RL}(\Sigma)} = \bigwedge_{p,q \in \mathrm{Sen}(\Sigma)} \mathrm{mp}(p,q)$.

The following result can be traced back to the many-valued interpretation of modus ponens from [Gog69].

PROPOSITION 5.2.7 (Many-valued modus-ponens). *For any Σ-sentences p and q, and any Σ-model M, if $(M \vDash_\Sigma p) \geq x$ and $(M \vDash_\Sigma p \to q) \geq y$, then $(M \vDash_\Sigma q) \geq x * y$.*

$$\frac{M \vDash_\Sigma p \geq x \qquad\qquad M \vDash_\Sigma p \to q \geq y}{M \vDash_\Sigma q \geq x * y}$$

PROOF. We know that, for any any residuated lattice $\mathcal{L}$, $x * (x \to y) \leq y$ holds for any $x, y \in L$. Thus, for every Σ-model M,

$$(M \vDash_\Sigma p) * ((M \vDash_\Sigma p) \to (M \vDash_\Sigma q)) \leq (M \vDash_\Sigma q).$$

Moreover,

$$(M \vDash_\Sigma p) * ((M \vDash_\Sigma p) \to (M \vDash_\Sigma q)) = (M \vDash_\Sigma p) * (M \vDash_\Sigma p \to q),$$

and because $*$ is monotonic in both arguments, $(M \vDash_\Sigma p) * (M \vDash_\Sigma p \to q) \geq x * y$. Therefore, $(M \vDash_\Sigma q) \geq x * y$. □

PROPOSITION 5.2.8 (Soundness of resolution). *Let $\langle \Sigma, E, \Gamma, \exists X_1.\alpha \wedge Q_i \wedge \omega \rangle$ be a problem and $\exists Y.\psi_1(\alpha) \wedge \psi_2(\overline{H}) \wedge \psi_1(\omega)$ be a query derived by resolution with* *a degree of confidence δ from $\exists X_1.\alpha \wedge Q_i \wedge \omega$ and the η-correct clause $\forall X_2.\overline{H} \to C$ using the unifier $\langle \psi_1 \colon X_1 \to Y, \psi_2 \colon X_2 \to Y \rangle$. Then for any ζ-solution $\psi \colon Y \to Y_1$ to the problem $\langle \Sigma, E, \Gamma, \exists Y.\psi_1(\alpha) \wedge \psi_2(\overline{H}) \wedge \psi_1(\omega)\rangle$, the substitution $\psi_1 \mathbin{;} \psi$ is a $(\zeta * \eta * \mathrm{mp} * \delta)$-solution to the initial problem, $\langle \Sigma, E, \Gamma, \exists X_1.\alpha \wedge Q_i \wedge \omega\rangle$.*

PROOF. From the fact that $\psi \colon Y \to Y_1$ is a ζ-solution to the problem $\langle \Sigma, E, \Gamma, \exists Y.\psi_1(\alpha) \wedge \psi_2(\overline{H}) \wedge \psi_1(\omega)\rangle$, we know that

$$(E \models_\Sigma \forall Y_1.\psi(\psi_1(\alpha) \wedge \psi_2(\overline{H}) \wedge \psi_1(\omega))) \geq \zeta.$$

From the generalization meta-rule, we have that

$$(E \models_{\Sigma(Y_1)} \psi(\psi_1(\alpha) \wedge \psi_2(\overline{H}) \wedge \psi_1(\omega))) \geq \zeta,$$

and in particular,

$$(E \models_{\Sigma(Y_1)} (\psi_2 \mathbin{;} \psi)(\overline{H}) \geq \zeta. \tag{5.8}$$

On the other hand, we have that $(E \models_\Sigma \forall X_2.\overline{H} \to C) \geq \eta$, and from the meta-rule of generalization, it follows that $(E \models_{\Sigma(X_2)} \overline{H} \to C) \geq \eta$. By applying the substitution $\psi_2 \mathbin{;} \psi$, we obtain

$$(E \models_{\Sigma(Y_1)} (\psi_2 \mathbin{;} \psi)(\overline{H}) \to (\psi_2 \mathbin{;} \psi)(C)) \geq \eta. \tag{5.9}$$

From (5.8) and (5.9), we have that

$$E \models_{\Sigma(Y_1)} (\psi_2 \mathbin{;} \psi)(\overline{H}) \wedge ((\psi_2 \mathbin{;} \psi)(\overline{H}) \to (\psi_2 \mathbin{;} \psi)(C)) \geq \zeta * \eta, \tag{5.10}$$

and we know that

$$((\psi_2 \mathbin{;} \psi)(\overline{H}) \wedge ((\psi_2 \mathbin{;} \psi)(\overline{H}) \to (\psi_2 \mathbin{;} \psi)(C))) \to (\psi_2 \mathbin{;} \psi)(C) \geq \mathrm{mp}. \tag{5.11}$$

By the rule from Proposition 5.2.7 for (5.10) and (5.11),

$$(E \models_{\Sigma(Y_1)} (\psi_2 \mathbin{;} \psi)(C)) \geq \eta * \zeta * \mathrm{mp}. \tag{5.12}$$

From the entailment of unification, we have that $\psi_2(C) \models_{\Sigma(Y)} \psi_1(Q_i) \geq \delta$, and thus, $(\psi_2 \mathbin{;} \psi)(C) \models_{\Sigma(Y_1)} (\psi_1 \mathbin{;} \psi)(Q_i) \geq \delta$. From (5.12) it follows that $E \models_{\Sigma(Y_1)} (\psi_1 \mathbin{;} \psi)(Q_i) \geq \eta * \zeta * \mathrm{mp} * \delta$. Because $(E \models_{\Sigma(Y_1)} (\psi_1 \mathbin{;} \psi)(\alpha \wedge \omega) \geq \zeta$, we have that $E \models_{\Sigma(Y_1)} (\psi_1 \mathbin{;} \psi)(\alpha \wedge Q_i \wedge \omega) \geq \eta * \zeta * \mathrm{mp} * \delta$. □

Notice the difference between Proposition 5.2.8 and Proposition 4.1.11 in the degree of confidence of the solutions that they provide. Because the set of sentences E is empty in the context of Proposition 4.1.11, we can obtain a proof where the value of the modus ponens does not contribute to the confidence of the solution. It is also worth mentioning that the soundness of service binding is a result concerning the preservation of the satisfiability of a given aplication by the binding process. In the more general context of many-valued logic programming, that kind of soundness is proved in two steps: the first one of an operational nature – the soundness of obtaining a solution through resolution as in Proposition 5.2.8 – and the second of a denotational nature – which makes the connection between solutions and satisfiability, as in Proposition 5.1.22.

COMPUTING SOLUTIONS TO INSTANCES OF PROBLEMS

We focus now on the definition of resolution for program instances, where we are interested in computing solutions to a problem with a degree of confidence greater than a predefined value.

DEFINITION 5.2.9. Given an arbitrary residuated lattice $\mathcal{L}$, we define $\dagger$ to be a binary operation on $\mathcal{L}$ such that

$$x \dagger y = \bigwedge \{z \in \mathrm{L} \mid x \leq y * z\}.$$

PROPOSITION 5.2.10. *For any elements x and y in a residuated lattice $\mathcal{L}$ with $\dagger$ defined as above, we have $x \leq y * (x \dagger y)$.*

PROOF. From the definition of $\dagger$,

$$y * (x \dagger y) = y * \bigwedge \{z \in \mathrm{L} \mid x \leq y * z\}$$

and from the distributivity of $*$ over arbitrary meets,

$$y * \bigwedge \{z \in \mathrm{L} \mid x \leq y * z\} = \bigwedge \{y * z \in \mathrm{L} \mid x \leq y * z\} \geq x.$$

□

PROPOSITION 5.2.11. *For any residuated lattice $\mathcal{L}$, $\dagger$ is the left adjoint of $*$.*

PROOF. We have to show that $x \dagger y \leq w \quad$ iff $\quad x \leq y * w$, for any $x, y, w \in \mathrm{L}$.

- Assume $x \dagger y \leq w$. Since $x \leq y * (x \dagger y)$ by Proposition 5.2.10, and since $*$ is monotonic, we obtain that $x \leq y * w$.
- Suppose $x \leq y * w$ and consider the set $Z = \{z \in \mathrm{L} \mid x \leq y * z\}$. Clearly, $w \in Z$. It follows that $w \geq \bigwedge Z = x \dagger y$.

□

DEFINITION 5.2.12 (Resolution for problem instances). Let $\langle \Sigma, E, \Gamma, \exists X_1.\alpha \wedge Q_i \wedge \omega, \mu \rangle$ be an instance of a many-valued logic-programming problem, with α and ω conjunctions of atoms, and $\forall X_2.\overline{H} \rightarrow C$ a η-correct Σ-clause from Γ. A Σ-query $\exists Y.\psi_1(\alpha) \wedge \psi_2(\overline{H}) \wedge \psi_1(\omega)$ is said to be *derived by resolution* with a degree of confidence $\mu \dagger (\eta * \mathrm{mp} * \delta)$ from $\exists X_1.\alpha \wedge Q_i \wedge \omega$ and $\forall X_2.\overline{H} \rightarrow C$ using the unifier $\langle \psi_1 \colon X_1 \rightarrow Y, \psi_2 \colon X_2 \rightarrow Y \rangle$ if $(\psi_2(C) \vDash_{\Sigma(Y)} \psi_1(Q_i)) \geq \delta$.

$$\frac{\langle \Sigma, E, \Gamma, \exists X_1.\alpha \wedge Q_i \wedge \omega, \mu \rangle \qquad \forall X_2 \cdot C \leftarrow \overline{H} \in \Gamma}{\langle \Sigma, E, \Gamma, \exists Y \cdot \psi_1(\alpha) \wedge \psi_2(\overline{H}) \wedge \psi_1(\omega), \mu \dagger (\eta * \mathrm{mp} * \delta) \rangle} \langle \psi_1, \psi_2 \rangle$$

PROPOSITION 5.2.13 (Soundness of resolution). *Let* $\langle \Sigma, E, \Gamma, \exists X_1.\alpha \wedge Q_i \wedge \omega, \mu \rangle$ *be an instance of a many-valued logic-programming problem, with* α *and* ω *conjunctions of atoms, and* $\exists Y.\psi_1(\alpha) \wedge \psi_2(\overline{H}) \wedge \psi_1(\omega)$ *be a query derived by resolution with a degree of confidence* $\mu \dagger (\eta * \mathrm{mp} * \delta)$ *from* $\exists X_1.\alpha \wedge Q_i \wedge \omega$ *and the* η*-correct clause* $\forall X_2.\overline{H} \rightarrow C$ *using the unifier* $\langle \psi_1 \colon X_1 \rightarrow Y, \psi_2 \colon X_2 \rightarrow Y \rangle$*. Then for any solution* $\psi \colon Y \rightarrow Y_1$ *to the instance* $\langle \Sigma, E, \Gamma, \exists Y.\psi_1(\alpha) \wedge \psi_2(\overline{H}) \wedge \psi_1(\omega), \mu \dagger (\eta * \mathrm{mp} * \delta) \rangle$*, the substitution* $\psi_1 \mathbin{;} \psi$ *is a solution to the initial instance,* $\langle \Sigma, E, \Gamma, \exists X_1.\alpha \wedge Q_i \wedge \omega, \mu \rangle$*.*

PROOF. Because $\psi \colon Y \rightarrow Y_1$ is a solution to the instance $\langle \Sigma, E, \Gamma, \exists Y.\psi_1(\alpha) \wedge \psi_2(\overline{H}) \wedge \psi_1(\omega), \mu \dagger (\eta * \mathrm{mp} * \delta) \rangle$, it means that ψ is a ζ-solution to the problem $\langle \Sigma, E, \Gamma, \exists Y.\psi_1(\alpha) \wedge \psi_2(\overline{H}) \wedge \psi_1(\omega) \rangle$ with $\zeta \geq \mu \dagger (\eta * \mathrm{mp} * \delta)$. From the soundness of resolution for a problem (Proposition 5.2.8), we know that $\psi_1 \mathbin{;} \psi$ is a ξ-solution to the initial problem $\langle \Sigma, E, \Gamma, \exists X_1.\alpha \wedge Q_i \wedge \omega \rangle$ with $\xi = \zeta * \eta * \mathrm{mp} * \delta$. All we need to show is that $\xi \geq \mu$, which follows easily from Proposition 5.2.10 since we have that $(\eta * \mathrm{mp} * \delta) * \zeta \geq (\eta * \mathrm{mp} * \delta) * (\mu \dagger (\eta * \mathrm{mp} * \delta)) \geq \mu$. □

5.3 MANY-VALUED: TRUE OR FALSE.

The connection between solving problems and solving problem instances belongs to a broader phenomenon of encoding (aspects) of many-valued logic into Boolean logic. This is grounded on the observation that the solutions to problems from Definition 5.1.20 are graded, whereas the solutions to problem instances from Definition 5.1.21 are Boolean. To explain this relationship in rigourous terms, we look at how arbitrary $\mathbb{RL}$-institutions can be 'reduced' to ordinary Boolean institutions. For the purpose of this section, it suffices to look at this relationship at the level of $\mathbb{RL}$-rooms and rooms. This relationship reminds of the MV-institutions introduced in [Dia13], hence the name of the following construction.

FACT 5.3.1. Any $\mathbb{RL}$-room $\mathcal{R} = \langle S, \mathbb{M}, \mathcal{L}, \vDash \rangle$ can be mapped to a Boolean room $\mathrm{MV}(\mathcal{R}) = \langle S^{\mathrm{MV}}, \mathbb{M}^{\mathrm{MV}}, \vDash^{\mathrm{MV}} \rangle$, where S^{MV} consists of pairs $\langle \rho, x \rangle$ of sentences ρ from S and values $x \in \mathcal{L}$, $\mathbb{M}^{\mathrm{MV}} = \mathbb{M}$, and the Boolean satisfaction of a sentence $\langle \rho, x \rangle$ by a model M is given by $M \vDash^{\mathrm{MV}} \rho \geq x$.

In fact, in all the 'many-valued' approaches to logic programming in the literature (that we know of), the underlying logic is Boolean, and can be seen as a functor that maps signatures to Boolean, MV-rooms obtained as above. For this reason, the logic-programming technology is actually a classical, Boolean one; due to the use of lattice elements within MV)-sentences, the denotational semantics, which is based on a notion of Boolean entailment, lacks some of the subtleties of graded consequence. As for the operational aspect, resolution is defined as follows:

DEFINITION 5.3.2 (Resolution for MV-problems). Let $\langle \Sigma, \Gamma, \exists X_1.\alpha \wedge (Q_i, \mu) \wedge \omega, \rangle$ be an MV-logic-programming problem, with α and ω conjunctions of atoms, and $(\forall X_2.\overline{H} \to C, \eta)$ a Σ-clause from Γ. A Σ-query $\exists Y.\psi_1(\alpha) \wedge (\psi_2(\overline{H}), \mu \dagger (\eta * \delta)) \wedge \psi_1(\omega)$ is said to be *derived by resolution* from $\exists X_1.\alpha \wedge (Q_i, \mu) \wedge \omega$ and $(\forall X_2.\overline{H} \to C, \eta)$ using the unifier $\langle \psi_1 \colon X_1 \to Y, \psi_2 \colon X_2 \to Y \rangle$ if $(\psi_2(C), 1) \vDash_{\Sigma(Y)} (\psi_1(Q_i), \delta)$.

$$\frac{\exists X_1.\alpha \wedge (Q_i, \mu) \wedge \omega \qquad (\forall X_2 \cdot C \leftarrow \overline{H}, \eta) \in \Gamma}{\exists Y \cdot \psi_1(\alpha) \wedge (\psi_2(\overline{H}), \mu \dagger (\eta * \delta)) \wedge \psi_1(\omega)} \; \langle \psi_1, \psi_2 \rangle$$

Notice the similarity between the resolution for MV-problems and the resolution for instances of many-valued logic programs from Definition 5.2.12. The factor mp is, of course, missing from the degree of confidence of the resulting query, as the sentence entailment in this setting is Boolean, and mp_2 is 1. Also, the existing MV-approaches to logic programming consider the classical, stricter notion of unifier, and thus δ is also 1.

COMPARING MANY-VALUEDNESS

PROPOSITION 5.3.3. *Let $\mathcal{R} = \langle S, \mathbb{M}, \mathcal{L}, \vDash \rangle$ be an $\mathbb{RL}$-room, ρ a sentence from S, and E a set of sentences from S. If $(E \vDash \rho) \geq y \dagger x$, then $\langle E, x \rangle \vDash^{\mathrm{MV}} (\rho, y)$.*

PROOF. Let us assume that $(E \vDash \rho) \geq y \dagger x$. For any model $M \in |\mathbb{M}|$ such that $(M \vDash E) \geq x$, we have by hypothesis that $((M \vDash E) \to (M \vDash \rho)) \geq y \dagger x$. By the rule of modus ponens from Proposition 5.2.7, $(M \vDash \rho) \geq x * (y \dagger x) \geq y$. □

PROPOSITION 5.3.4. *The converse of the implication in Proposition 5.3.3 does not hold in general.*

PROOF. Let us assume we are working under the many-valued relational first-order logic $\mathcal{REL}_{\mathbb{RL}}$, that $\mathcal{L}$ is the Łukasiewicz lattice $\mathcal{L}_3$, E is the set $\{p, p \to q\}$ with p and q nullary predicates and $\rho = q$, and x, y are equal to 1. If $M \in |\mathbb{M}|$ is a model that satisfies E with the value 1, then from the modus ponens meta-rule for $\mathrm{MV}(\mathcal{R})$, it also satisfies ρ with the value $1 * 1 = 1$. Therefore, $\langle E, x \rangle \models^{\mathrm{MV}} (\rho, y)$. On the other hand, $E \models \rho$ coincides with the value of modus ponens for $\mathcal{L}_3$, which is known to be $\mathrm{mp}_{\mathcal{L}_3} = 0.5$. We thus obtain $E \models \rho = 0.5 \ngeq y \dagger x = 1$. □

FUTURE DIRECTIONS

The results above raise the question of adequacy of the two frameworks, as they are clearly not equivalent for any underlying lattice. One could interpret the results in a different, more positive manner: when using only Heyting algebras, for example, for which the value of modus ponens is always 1 (as for the Boolean case), any many-valued logic-programming problem can be reduced to its Boolean, MV correspondent, thus enjoying the benefits of working in a well-studied framework for logic programming. More specifically, we obtain for free tool support, and in addition, a way to ensure completeness results for the kind of many-valued logic-programming that we have considered in this chapter. For the latter, it is worth noting that the issue of completeness for this kind of logic programming is open due to factors that extend on several levels.

At a very basic level, which corresponds to the proof theory of the underlying institution, it implies the development of a complete syntactical method for the calculation of graded consequence. To the best of our knowledge, this is a new field, which hasn't received much attention.

On a second level, we have to ensure that the satisfiability of a problem corresponds to the existence of an adequate solution to the problem. This would corrrepond to the converse of Proposition 5.1.22, and depends on the nature of substitutions. This falls outside the scope of this thesis, for which substitutions are purely semantic.

Lastly, at the third, operational level, we would have to ensure that any solution can be discovered through resolution. This is where completeness fails, because we rely on a notion of *best unifier*, which, as opposed to the classical notion of most general unifier, does not subsume other unifiers/solutions. To address this, independently from the operational semantics, we would have to further study those solutions that are maximal in the sense that their degrees of confidence are maximal. Along this line of research, as future work, we aim to examine the issue of completeness relatively to the class of maximal solutions.

6 CONCLUSIONS

6.1 CREATIVE SYSTEMS

In this thesis, we have shown how aspects of service-oriented computing can be used to model a particular kind of computational creativity, namely the one that arises from exploratory creative systems. These are systems in which creativity is expressed as the discovery/identification of new concepts in a predefined conceptual space. Very roughly, this kind of creativity resembles the traversal of a graph whose nodes are concepts, and whose links correspond to concept discovery.

Concretely, the main contribution of this thesis is to provide a framework that accounts for rigorous definitions of creative systems, including concepts, conceptual spaces, and most importantly, the strategy through which a conceptual space is traversed. The following analogy between creativity, logic (in particular logic programming), and service-oriented computing is an accurate depiction of the developments in this thesis:

- The *language* used to describe concepts, a notion that is specific to computational creativity, corresponds in logic to the choice of a suitable *logical formalism*. Traditionally, researchers in computational creativity have focused on fragments of first-order logic. In this regard, we eliminate this restriction by replacing first-order logic with an arbitrary institution and by lifting to the institutional level those properties of first-order logic that are used in creative systems. In service-oriented computing, the language via this abstractization to an institution can be seen as a *specification/programming language* for services.
- Following a trending relationship with algebraic specification, *concepts* correspond to presentations over the logical system defined above. We regard these presentations as signatures over a richer logical system, which enables us to establish a second connection with service-oriented computing: concepts correspond to *orchestrations* of service applications or modules.

- The *conceptual space* in computational creativity, which limits the scope of the exploration of concepts, corresponds in logic to the *Herbrand universe* for a fixed presentation regarded as ground. In service-oriented computing, this corresponds to a shared *API, or data layer*, between service applications and modules – that is what allows applications and modules to be accepted as entities of the same computational environment by a service-oriented middleware.
- The *strategy for the traversal of the conceptual space* was initially a classical state-space search based on backtracking (as in artificial intelligence); later it evolved into a more general category of methods or techniques that also includes evolutionary algorithms. From a logical perspective, the strategy corresponds to operational proof techniques. The approach that we consider is focused on the use of *resolution* (specific to logic programming), as a way to discover new concepts with the help of user-defined rules. This requires a further level of clarification:
  - we wrap concepts (presentations) into logic-programming queries, which extend concepts by adding so-called requirements through which concepts relate to other concepts;
  - we capture rules as logic-programming clauses, which have the role of morphing queries into other queries, whose underlying concepts are generally more complex than the original ones, and whose requirements are simpler.

  In the context of service-oriented computing, by further exploring the analogy with logic programming, the queries correspond to service applications, the logic-programming clauses to service modules, and resolution to the processes of *service discovery and binding*.
- Lastly, the *evaluation of concepts*, which is meant to determine their degree of novelty and usefulness (with respect to given criteria) corresponds in logic to a semantic entailment between the criteria (formalized as a set of sentences) and the concept to be evaluated (which is given by a set of sentences as well). In the literature, this kind of entailment is usually considered in a Boolean sense; for creativity, a graded (many-valued) view on entailment is much more useful. This forms the basis of our notion of *unifier*. For services, the evaluation of concepts and the unification correspond to the degree of the *fulfilment of service level agreements* between service modules and applications.

The analogy outlined above gives a full treatment of exploratory creativity. Although we did not focus on transformational creativity, we briefly sketch a possible extension of our model that would accommodate transformational creative systems as well. This type of creativity is born from the change of the limits of the conceptual space to include new concepts. At the logical level of our model, we would need to study the change of the ground signature over which we define logic programs, and the way in which such a signature morphism would influence resolution and unification. Here lies one of the main benefits of the logic-programming semantics of service-oriented computing and creative systems that we explored in the last chapter of the thesis: in logic-programming, this change of the ground signature can be accommodated through a so-called view (morphism) between logic programs. This notion is at the core of the modularization of logic programs, a field which has been studied extensively over the past decades. However, the great majority of the literature deals with logic programs that are Boolean, not graded, as they would need to be for creative systems. This opens the possibility of a new line of research, beyond the scope of this thesis, into graded logic-programming – not to be confused, as we have seen in Section 5.3, with many-valued or fuzzy logic-programming, which have already received adequate treatments.

As argued, dealing with creativity through the filter of logic programming can lead to a framework that integrates both exploratory and transformational creativity. The same cannot be said however about combinational creativity, which deals with the creation of new concepts without relying on rules (as in exploratory creativity) nor on changes of the state space (as in transformational creativity). Instead, it relies on a key notion of blending of concepts, so it operates directly at the level of conceptual spaces. In what follows, we outline some of the key ideas that underlie combinational creativity and hint at the way in which the research that we presented in this thesis can be adapted to support blending.

6.2 COMBINATIONAL CREATIVITY

CONCEPTUAL SPACES. Mental spaces, or conceptual spaces, as defined by Fauconnier in [FSL94], are a formalization of the idea that concepts are used not in isolation, but in groups of related concepts. The focus is not on concepts themselves, as mental spaces abstract over the actual properties of concepts, but on the relationships that develop between them. This formalization is based on relational first-order logic, using constants and binary relations to represent concepts and the relations between them.

In more recent approaches to conceptual spaces, such as the one of

Goguen introduced in [Gog99], the logic of order-sorted algebra[1] [GM92] is chosen as the mathematical foundation of representing concepts (and implicitly the conceptual spaces they belong to), by combining both the idea of inheritance and the one of mereology structure (whole/part), through a tractable algebraic formalism. The choice of this formalism is also motivated by its appropriateness to modelling concept blending.

CONCEPTUAL BLENDING. Conceptual blending is a fundamental cognitive operation of obtaining a new conceptual space from two different spaces. As introduced by Fauconnier and Turner in [FT96], conceptual blending is the combination of parts of two or more input conceptual spaces into a new space. The intuition comes from natural language, with the very common blends of words, like *houseboat* (the most traditional example in the field) and *computer virus*, and with many types of metaphors. The simplest scheme of such a blend is given in the diagram below, where I_1 and I_2 are two input spaces, G is a space that contains so-called generic elements of the input spaces, the maps from G to I_1 and I_2 indicate what elements should be identified, and B is the new conceptual space, or the *blend*:

$$\begin{tikzcd}
& B & \\
I_1 \arrow[ru] & & I_2 \arrow[lu] \\
& G \arrow[lu] \arrow[ru] \arrow[uu] &
\end{tikzcd}$$

FIGURE 6.1. Concept blending

It is now apparent that, if one uses presentations over a logic such as order-sorted algebra (as Goguen did), the construction above is a special case of a categorical colimit – a pushout. However, in contrast with the notion of colimit, blends are not determined uniquely up to isomorphisms.

Let us consider the following example (non-classical in that it does not involve houses or boats): we start with the input concepts of a jumper illustrated in Figure 6.2 and of a baby from Figure 6.3. A jumper is characterized by the material it is made of – wool, and the entity that wears it – a person. A baby is characterized by its belonging to a mother, and by being made of sugar, spice, and everything nice, of course. We assume all these are

[1] In essence, order-sorted algebra extends the equational logic from Example 2.2.8 with a partial order on the sets of sorts, which then translates to subset inclusions between the carrier sets of those sorts.

adequately specified using order-sorted algebra – which is straightforward to do.

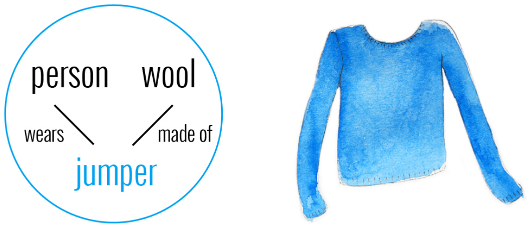


FIGURE 6.2. The concept of a jumper

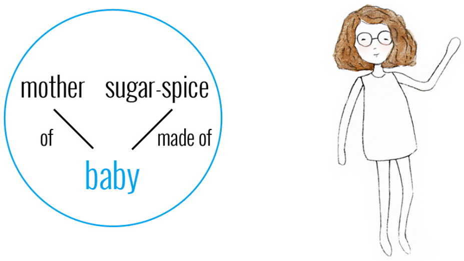


FIGURE 6.3. The concept of a baby

There is obviously more than just one possible combination of these two concepts – provided that we ignore the formal definition of a categorical colimit. Some of these possible blends are illustrated in Figure 6.4, as a concept of a jumper that belongs to a mother (made of wool and worn by a mother), in Figure 6.5, as a concept of a baby jumper (made of wool and belonging to a mother jumper), and finally in Figure 6.6, as a less common blend of a baby wearing a jumper.

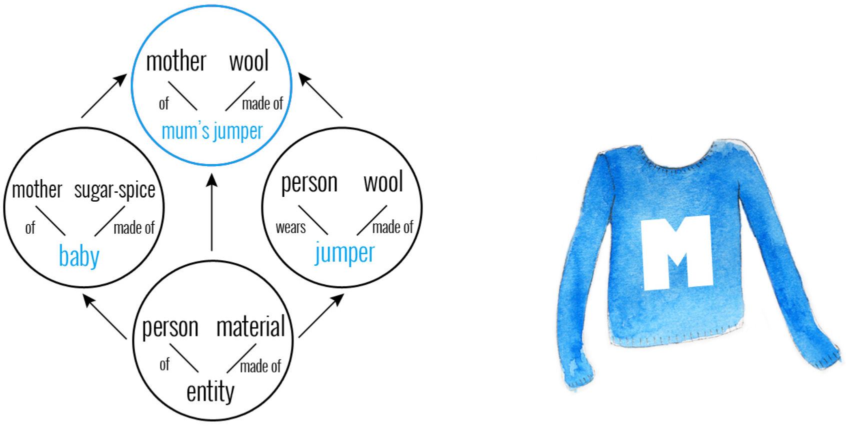


FIGURE 6.4. The concept of a mother's jumper

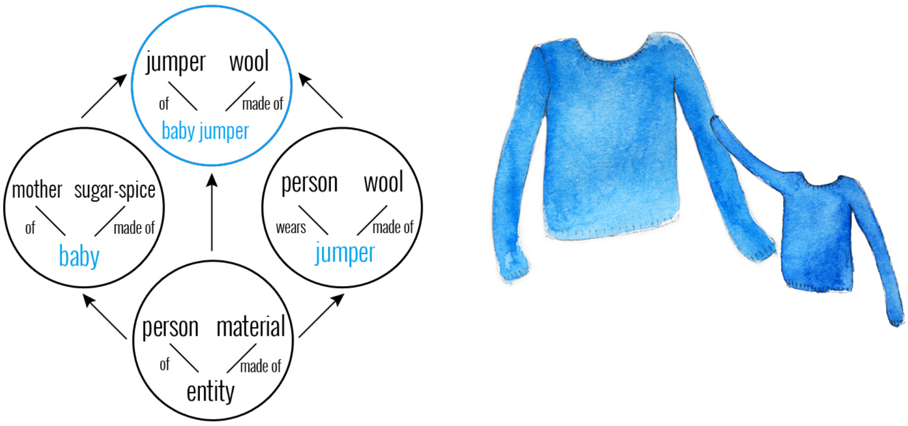


FIGURE 6.5. The concept of a baby jumper

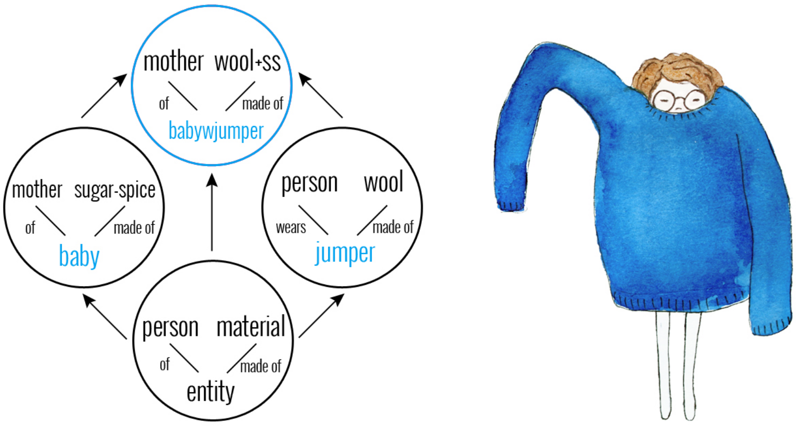


FIGURE 6.6. The concept of a baby wearing a jumper

To allow for more blends to be obtained through a categorical construct similar to a colimit, Goguen introduced in [Gog99] the concept of a 3/2-category, which is is a category $\mathbb{C}$ such that for every two objects A and B, the set $\mathbb{C}(A, B)$ is partially ordered, composition preserves the orderings, and identities are maximal.

More precisely, for two morphisms of conceptual spaces $G \longrightarrow I_1$ and $G \longrightarrow I_2$, a blend is defined as a space B together with morphisms $I_1 \longrightarrow B$, $I_2 \longrightarrow B$, and $G \longrightarrow B$ (which are called injections), such that the diagram in Figure 6.1 weakly commutes. This means that both the compositions $G \longrightarrow I_1 \longrightarrow B$ and $G \longrightarrow I_2 \longrightarrow B$ are weakly equal to the morphism $G \longrightarrow B$, i.e. each symbol in G is mapped to the same symbol in B under them, provided that the morphism is defined on the symbol. When $G \longrightarrow B$ is totally defined, the compositions $G \longrightarrow I_1 \longrightarrow B$ and $G \longrightarrow I_2 \longrightarrow B$ are weakly equal.

In order to evaluate the quality of the possible blends and to choose one of them, some optimality principles can be imposed, but these are left to specifier, and are not provided by the current formulation of the framework.

BEYOND THE STATE OF THE ART. Recently, Diaconescu has suggested a different approach to the problem: adapting the concept of institution to naturally accommodate the notion of 3/2-signature pushouts [Dia17]. The result of that study, 3/2-institutions, extends the classical notion of institution by considering that, for any two fixed signatures Σ and Ω, the collection of signature morphisms between them is not just a set, but a partially ordered set. This has multiple implications on the semantics of such institutions, the most important of which is that it makes 3/2-institutions suitable for dealing with creative phenomena, beyond the modelling of concept blending.

In particular, one of the main aspects that we would like to pursue next is building a framework based on 3/2-institutions that would allow modelling all three types of creativity: concepts would be defined as presentations over such an institution, concept blending would be captured by the calculation of a 3/2-colimit (instead of an ordinary colimit as we did when binding services to applications), and the exploratory and transformational creativity would be simulated through a mechanism based on the one that we have defined in Chapters 4 and 5.

BIBLIOGRAPHY

[AHS09] Jirí Adámek, Horst Herrlich and George E. Strecker. *Abstract and Concrete Categories - The Joy of Cats*. Dover Publications, 2009 (cited on pages 13, 32).

[AD07] Marc Aiguier and Răzvan Diaconescu. 'Stratified institutions and elementary homomorphisms'. In: *Information Processing Letters* 103.1 (2007), pages 5–13 (cited on page 51).

[AM11] Torsten Anders and Eduardo Reck Miranda. 'Constraint programming systems for modeling music theories and composition'. In: *ACM Comput. Surv.* 43.4 (2011), page 30 (cited on page 44).

[BD74] Raymond Balbes and Philip Dwinger. *Distributive lattices*. University of Missouri press, 1974 (cited on page 27).

[Bes+17a] Tarek R. Besold et al. 'Neural-Symbolic Learning and Reasoning: A Survey and Interpretation'. In: *CoRR* abs/1711.03902 (2017). URL: http://arxiv.org/abs/1711.03902 (cited on page 9).

[Bes+17b] Tarek R. Besold et al. 'Reasoning in Non-probabilistic Uncertainty: Logic Programming and Neural-Symbolic Computing as Examples'. In: *Minds and Machines* 27.1 (2017), pages 37–77 (cited on page 102).

[BG06] Stefano Bistarelli and Fabio Gadducci. 'Enhancing Constraints Manipulation in Semiring-Based Formalisms'. In: *ECAI 2006*. Edited by Gerhard Brewka et al. Volume 141. IOS Press, 2006, pages 63–67 (cited on page 48).

[BMR97] Stefano Bistarelli, Ugo Montanari and Francesca Rossi. 'Semiring-based constraint satisfaction and optimization'. In: *Journal of the ACM* 44.2 (1997), pages 201–236 (cited on page 45).

[BS09] Stefano Bistarelli and Francesco Santini. 'A Nonmonotonic Soft Concurrent Constraint Language for SLA Negotiation'. In: *Electr. Notes Theor. Comput. Sci.* 236 (2009), pages 147–162 (cited on page 82).

[Bis+99] Stefano Bistarelli et al. 'Semiring-based CSPs and valued CSPs: Frameworks, properties, and comparison'. In: *Constraints* 4.3 (1999), pages 199–240 (cited on pages 44, 48).

[BY04] Tim Blackwell and Michael Young. 'Self-organised music'. In: *Organised Sound* 9.02 (2004), pages 123–136 (cited on page 81).

[Bod91] Margaret A. Boden. *The Creative Mind: Myths and Mechanisms*. New York, NY, USA: Basic Books, Inc., 1991 (cited on pages 6, 7).

[Bor94] Francis Borceux. *Handbook of Categorical Algebra*. Volume 1. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 1994 (cited on page 16).

[Bor05] David Borgo. *Sync or swarm: Improvising music in a complex age*. A&C Black, 2005 (cited on pages 59, 66, 80, 82).

[BG05] David Borgo and Joseph A. Goguen. 'Rivers of consciousness: The nonlinear dynamics of free jazz'. In: *Jazz Research Proceedings Yearbook*. Volume 25. 2005 (cited on pages 59–63, 65, 66, 80).

[Bov09] Simone Bova. 'Soft Constraints Processing over Divisible Residuated Lattices'. In: *Symbolic and Quantitative Approaches to Reasoning with Uncertainty*. Edited by Claudio Sossai and Gaetano Chemello. Volume 5590. LNCS. Springer, 2009, pages 887–898 (cited on page 48).

[Bra88] Anthony Braxton. *Composition notes*. Volume 3. Synthesis Music, 1988 (cited on page 68).

[CVW09] Amílcar Cardoso, Tony Veale and Geraint A. Wiggins. 'Converging on the Divergent: The History (and Future) of the International Joint Workshops in Computational Creativity'. In: *AI Magazine* 30.3 (2009), pages 15–22 (cited on page 6).

[Cha88] Mihir K. Chakraborty. 'Use of fuzzy set theory in introducing graded consequence in multiple valued logic'. In: *Fuzzy logic in knowledge-based systems, Decision and Control*. Edited by MM Gupta and T Yamakawa. Elsevier Science, 1988, pages 247–257 (cited on page 35).

[Cha95] Mihir K. Chakraborty. 'Graded consequence: further studies'. In: *Journal of Applied Non-Classical Logics* 5.2 (1995), pages 227–238 (cited on page 35).

[CF16] Claudia Elena Chiriţă and José Luiz Fiadeiro. 'Free Jazz in the Land of Algebraic Improvisation'. In: *Proceedings of the Seventh International Conference on Computational Creativity*. Edited by François Pachet et al. Sony CSL Paris, France, 2016 (cited on page 12).

[CFO16] Claudia Elena Chiriţă, José Luiz Fiadeiro and Fernando Orejas. 'Many-Valued Institutions for Constraint Specification'. In: *Fundamental Approaches to Software Engineering*. Volume 9633. LNCS. Springer, 2016, pages 359–376 (cited on page 12).

[Coh+03] David A. Cohen et al. 'Soft Constraints: Complexity and Multimorphisms'. In: *Principles and Practice of Constraint Programming*. Edited by Francesca Rossi. Volume 2833. LNCS. Springer, 2003, pages 244–258 (cited on page 45).

[CW12] Simon Colton and Geraint A. Wiggins. 'Computational Creativity: The Final Frontier?' In: *ECAI 2012 - 20th European Conference on Artificial Intelligence. Including Prestigious Applications of Artificial Intelligence (PAIS-2012) System Demonstrations Track, Montpellier, France, August 27-31 , 2012*. Edited by Luc De Raedt et al. Volume 242. Frontiers in Artificial Intelligence and Applications. IOS Press, 2012, pages 21–26 (cited on pages 3, 6).

[Icc] *Computational Creativity Manifesto*. http://computationalcreativity.net/home/about/computational-creativity/ (cited on page 6).

[Dia95] Răzvan Diaconescu. 'Completeness of category-based equational deduction'. In: *Mathematical Structures in Computer Science* 5.1 (1995), pages 9–40 (cited on page 18).

[Dia02] Răzvan Diaconescu. 'Grothendieck institutions'. In: *Applied Categorical Structures* 10.4 (2002), pages 383–402 (cited on page 39).

[Dia04] Răzvan Diaconescu. 'Herbrand theorems in arbitrary institutions'. In: *Information Processing Letters* 90.1 (2004), pages 29–37 (cited on page 102).

[Dia08] Răzvan Diaconescu. *Institution-independent Model Theory*. Studies in Universal Logic. Springer London, Limited, 2008. ISBN: 9783764387082 (cited on pages 18, 22).

[Dia10] Răzvan Diaconescu. 'Quasi-boolean encodings and conditionals in algebraic specification'. In: *Journal of Logic and Algebraic Programming* 79.2 (2010), pages 174–188 (cited on page 105).

[Dia11] Răzvan Diaconescu. 'Grothendieck Inclusion Systems'. In: *Applied Categorical Structures* 19.5 (2011), pages 783–802 (cited on pages 16, 84).

[Dia13] Răzvan Diaconescu. 'Institutional semantics for many-valued logics'. In: *Fuzzy Sets and Systems* 218 (2013), pages 32–52 (cited on pages 18, 105, 119).

[Dia14] Răzvan Diaconescu. 'Graded consequence: an institution theoretic study'. In: *Soft Comput.* 18.7 (2014), pages 1247–1267 (cited on pages 34, 35, 38, 57, 67).

[Dia17] Răzvan Diaconescu. '3/2-Institutions: an institution theory for conceptual blending'. In: *ArXiv e-prints* (2017). arXiv: 1708.09675 (cited on page 128).

[DGS93] Răzvan Diaconescu, Joseph A. Goguen and Petros Stefaneas. 'Logical support for modularisation'. In: *Logical Environments*. Edited by Gérard Huet and Gordon Plotkin. Cambridge University Press, 1993, pages 83–130 (cited on pages 22, 58, 104).

[DM16] Răzvan Diaconescu and Alexandre Madeira. 'Encoding hybridized institutions into first-order logic'. In: *Mathematical Structures in Computer Science* 26.5 (2016), pages 745–788 (cited on page 18).

[DS07] Răzvan Diaconescu and Petros Stefaneas. 'Ultraproducts and possible worlds semantics in institutions'. In: *Theor. Comput. Sci.* 379.1-2 (2007), pages 210–230 (cited on page 18).

[Dil39] Robert Palmer Dilworth. 'Non-Commutative Residuated Lattices'. In: *Transactions of the American Mathematical Society* 46.3 (1939), pages 426–444 (cited on page 27).

[DBC13] Soma Dutta, Sanjukta Basu and Mihir K. Chakraborty. 'Many-Valued Logics, Fuzzy Logics and Graded Consequence: A Comparative Appraisal'. In: *Logic and Its Applications, 5th Indian Conference, ICLA 2013, Chennai, India, January 10-12, 2013. Proceedings*. Edited by Kamal Lodaya. Volume 7750. Lecture Notes in Computer Science. Springer, 2013, pages 197–209 (cited on page 34).

[EH10] Patrik Eklund and Robert Helgesson. 'Monadic extensions of institutions'. In: *Fuzzy Sets and Systems* 161.18 (2010), pages 2354–2368 (cited on page 35).

[Epp+15] Manfred Eppe et al. 'Computational Invention of Cadences and Chord Progressions by Conceptual Chord-Blending'. In: *IJCAI 2015*. AAAI Press, 2015, pages 2445–2451 (cited on page 61).

[FSL94] Gilles Fauconnier, Eve Sweester and George Lakoff. *Mental Spaces: Aspects of Meaning Construction in Natural Language*. Cambridge University Press, 1994 (cited on page 124).

[FT96] Gilles Fauconnier and Mark Turner. 'Blending as a central process of grammar'. In: *Conceptual Structure, Discourse and Language*. Edited by Adele E. Goldberg. CSLI, 1996, pages 113–129 (cited on page 125).

[Fia+11] José Fiadeiro et al. 'The Sensoria Reference Modelling Language'. In: *Rigorous Software Engineering for Service-Oriented Systems: Results of the SENSORIA Project on Software Engineering for Service-Oriented Computing*. Edited by Martin Wirsing and Matthias Hölzl. Springer Berlin Heidelberg, 2011, pages 61–114 (cited on page 93).

[Fia05] José Luiz Fiadeiro. *Categories for software engineering*. Springer, 2005 (cited on page 14).

[FC96] José Luiz Fiadeiro and José Félix Costa. 'Mirror, Mirror in my Hand: A Duality between Specifications and Models of Process Behaviour'. In: *Mathematical Structures in Computer Science* 6.4 (1996), pages 353–373 (cited on page 18).

[FL13a] José Luiz Fiadeiro and Antónia Lopes. 'A model for dynamic reconfiguration in service-oriented architectures'. In: *Software and System Modeling* 12.2 (2013), pages 349–367 (cited on pages 82, 97).

[FL13b] José Luiz Fiadeiro and Antónia Lopes. 'An interface theory for service-oriented design'. In: *Theor. Comput. Sci.* 503 (2013), pages 1–30 (cited on pages 81, 82, 97).

[FLA12] José Luiz Fiadeiro, Antónia Lopes and João Abreu. 'A formal model for service-oriented interactions'. In: *Sci. Comput. Program.* 77.5 (2012), pages 577–608 (cited on page 93).

[FLB06] José Luiz Fiadeiro, Antónia Lopes and Laura Bocchi. 'Algebraic Semantics of Service Component Modules'. In: *Recent Trends in Algebraic Development Techniques*. Edited by José Luiz Fiadeiro and Pierre-Yves Schobbens. Volume 4409. LNCS. Springer, 2006, pages 37–55 (cited on page 81).

[FLB11] José Luiz Fiadeiro, Antónia Lopes and Laura Bocchi. 'An abstract model of service discovery and binding'. In: *Formal Asp. Comput.* 23.4 (2011), pages 433–463 (cited on pages 81, 82).

[GP10] Daniel Găină and Marius Petria. 'Completeness by Forcing'. In: *J. Log. Comput.* 20.6 (2010), pages 1165–1186 (cited on page 74).

[GJ09] Nikolaos Galatos and Peter Jipsen. 'A survey of Generalized Basic Logic algebras'. In: *Witnessed Years: Essays in Honour of Petr Hajek*. Edited by P. Cintula, Z. Hanikova and V. Svejdar. College Publications, 2009, 305—331 (cited on pages 27, 30).

[Gal+07] Nikolaos Galatos et al. *Residuated Lattices: An Algebraic Glimpse at Substructural Logics*. Studies in Logic and the Foundations of Mathematics. Elsevier Science, 2007 (cited on pages 27, 32).

[Gog69] Joseph A. Goguen. 'The Logic of Inexact Concepts'. In: *Synthese* 19.3/4 (1969), pages 325–373 (cited on page 115).

[Gog99] Joseph A. Goguen. 'An introduction to algebraic semiotics, with application to user interface design'. In: *Computation for metaphors, analogy, and agents*. Springer, 1999, pages 242–291 (cited on pages 59, 61, 125, 127).

[GB83] Joseph A. Goguen and Rod Burstall. 'Introducing Institutions'. In: *Logics of Programs, Workshop, Carnegie Mellon University, Pittsburgh, PA, USA, June 6-8, 1983, Proceedings*. Edited by Edmund M. Clarke and Dexter Kozen. Volume 164. Lecture Notes in Computer Science. Springer, 1983, pages 221–256 (cited on page 17).

[GB85] Joseph A. Goguen and Rod M. Burstall. 'A Study in the Foundations of Programming Methodology: Specifications, Institutions, Charters and Parchments'. In: *Category Theory and Computer Programming, Tutorial and Workshop, Guildford, UK, September 16-20, 1985 Proceedings*. Edited by David H. Pitt et al. Volume 240. Lecture Notes in Computer Science. Springer, 1985, pages 313–333 (cited on pages 26, 35).

[GB92] Joseph A. Goguen and Rod M. Burstall. 'Institutions: Abstract Model Theory for Specification and Programming'. In: *Journal of the ACM* 39.1 (1992), pages 95–146 (cited on pages 8, 17, 18, 25, 56).

[GM87] Joseph A. Goguen and José Meseguer. 'Models and equality for logical programming'. In: *Theory and Practice of Software Development*. Edited by Hartmut Ehrig et al. Volume 250. Lecture Notes in Computer Science. Springer, 1987, pages 1–22 (cited on page 18).

[GM92] Joseph A. Goguen and José Meseguer. 'Order-Sorted Algebra I: Equational Deduction for Multiple Inheritance, Overloading, Exceptions and Partial Operations'. In: *Theor. Comput. Sci.* 105.2 (1992), pages 217–273 (cited on page 125).

[GR02] Joseph A. Goguen and Grigore Roşu. 'Institution morphisms'. In: *Formal Aspects of Computing* 13.3–5 (2002), pages 274–307 (cited on page 23).

[Got01] Siegfried Gottwald. *A Treatise on Many-Valued Logics*. Studies in logic and computation. Research Studies Press, 2001 (cited on page 34).

[Háj98] Petr Hájek. *Metamathematics of Fuzzy Logic*. Volume 4. Trends in Logic. Springer, 1998 (cited on page 34).

[HKT01] David Harel, Dexter Kozen and Jerzy Tiuryn. 'Dynamic logic'. In: *Handbook of philosophical logic*. Springer, 2001, pages 99–217 (cited on page 67).

[HS73] H. Herrlich and G.E. Strecker. *Category theory: an introduction*. Allyn and Bacon series in advanced mathematics. Allyn and Bacon, 1973 (cited on page 15).

[HOT06] Geoffrey E. Hinton, Simon Osindero and Yee Whye Teh. 'A Fast Learning Algorithm for Deep Belief Nets'. In: *Neural Computation* 18.7 (2006), pages 1527–1554 (cited on page 9).

[Höh95] Ulrich Höhle. 'Commutative, residuated l-monoids'. In: *Non-Classical Logics and their Applications to Fuzzy Subsets: A Handbook of the Mathematical Foundations of Fuzzy Set Theory*. Edited by Ulrich Höhle and Erich Peter Klement. Springer, 1995, pages 53–106 (cited on page 27).

[HMW09] Matthias M. Hölzl, Max Meier and Martin Wirsing. 'Which Soft Constraints do you Prefer?' In: *Electr. Notes Theor. Comput. Sci.* 238.3 (2009), pages 189–205 (cited on page 82).

[Höl+09] Matthias M. Hölzl et al. 'Constraint-Muse: A Soft-Constraint Based System for Music Therapy'. In: *Algebra and Coalgebra in Computer Science, Third International Conference, CALCO 2009, Udine, Italy, September 7-10, 2009. Proceedings*. Edited by Alexander Kurz, Marina Lenisa and Andrzej Tarlecki. Volume 5728. Lecture Notes in Computer Science. Springer, 2009, pages 423–432 (cited on page 45).

[Idz84] Pawel Idziak. 'Lattice operation in BCK-algebras'. In: *Mathematica Japonica* 29 (1984), pages 839–846 (cited on pages 27, 29).

[JT02] Peter Jipsen and Constantine Tsinakis. 'A Survey of Residuated Lattices'. In: *Ordered Algebraic Structures: Proceedings of the Gainesville Conference Sponsored by the University of Florida 28th February – 3rd March, 2001*. Edited by Jorge Martínez. Springer, 2002, pages 19–56 (cited on page 27).

[KP+14] Maximos Kaliakatsos-Papakostas et al. 'Concept Invention and Music: Creating Novel Harmonies via Conceptual Blending'. In: *CIM2014*. 2014 (cited on page 61).

[KO] Tomasz Kowalski and Hiroakira Ono. *Residuated lattices: An algebraic glimpse at logics without contraction*. Technical report (cited on pages 27, 28).

[KLV04] Stanislav Krajci, Rastislav Lencses and Peter Vojtás. 'A comparison of fuzzy and annotated logic programming'. In: *Fuzzy Sets and Systems* 144.1 (2004), pages 173–192 (cited on page 102).

[Kru24] Wolfgang Krull. 'Axiomatische Begründung der allgemeinen Idealtheorie'. In: *Sitzungsberichte der physikalisch medizinischen Societät der Erlangen* 56 (1924), pages 47–63 (cited on page 27).

[Lan98] Saunders Mac Lane. *Categories for the Working Mathematician*. Springer, 1998 (cited on page 13).

[Llo87] John W. Lloyd. *Foundations of Logic Programming*. Symbolic computation: Artificial intelligence. Springer, 1987 (cited on page 111).

[Loc08] Graham Lock. 'What I Call a Sound: Anthony Braxton's Synaesthetic Ideal and Notations for Improvisers'. In: *Critical Studies in Improvisation* 4.1 (2008) (cited on pages 67, 68).

[Luk98] Thomas Lukasiewicz. 'Many-Valued First-Order Logics with Probabilistic Semantics'. In: *Computer Science Logic, 12th International Workshop, CSL '98, Annual Conference of the EACSL, Brno, Czech Republic, August 24-28, 1998, Proceedings*. Edited by Georg Gottlob, Etienne Grandjean and Katrin Seyr. Volume 1584. Lecture Notes in Computer Science. Springer, 1998, pages 415–429 (cited on page 102).

[Luk99] Thomas Lukasiewicz. 'Probabilistic and Truth-Functional Many-Valued Logic Programming'. In: *29th IEEE International Symposium on Multiple-Valued Logic, ISMVL 1999, Freiburg im Breisgau, Germany, May 20-22, 1999, Proceedings*. IEEE Computer Society, 1999, pages 236–241 (cited on page 102).

[Mal93] Grzegorz Malinowski. *Many-Valued Logics*. Oxford Logic Guides. Clarendon Press, 1993 (cited on page 93).

[May85] Brian H Mayoh. 'Galleries and institutions'. In: *DAIMI Report Series, Aarhus University* 14.191 (1985) (cited on page 35).

[MC08] Guerino Mazzola and Paul B Cherlin. *Flow, gesture, and spaces in free jazz: Towards a theory of collaboration*. Springer Science & Business Media, 2008 (cited on page 59).

[Mes89] José Meseguer. 'General Logics'. In: *Logic Colloquium'87 Proceedings of the Colloquium held in Granada*. Edited by H.-D. Ebbinghaus et al. Volume 129. Studies in Logic and the Foundations of Mathematics. Elsevier, 1989, pages 275 –329 (cited on page 23).

[MS90] Marcel Mongeau and David Sankoff. 'Comparison of musical sequences'. In: *Comput. Hum.* 24.3 (1990), pages 161–175 (cited on page 69).

[Mos02] Till Mossakowski. 'Comorphism-Based Grothendieck Logics'. In: *Mathematical Foundations of Computer Science 2002, 27th International Symposium, MFCS 2002, Warsaw, Poland, August 26-30, 2002, Proceedings*. Edited by Krzysztof Diks and Wojciech Rytter. Volume 2420. Lecture Notes in Computer Science. Springer, 2002, pages 593–604 (cited on pages 26, 39, 42).

[MML07] Till Mossakowski, Christian Maeder and Klaus Lüttich. 'The Heterogeneous Tool Set, Hets'. In: *TACAS*. Edited by Orna Grumberg and Michael Huth. Volume 4424. LNCS. Springer, 2007, pages 519–522 (cited on page 45).

[Mos04] Peter D. Mosses. *CASL Reference Manual*. Volume 2960. LNCS. Springer, 2004 (cited on page 28).

[NS92] Raymond T. Ng and V. S. Subrahmanian. 'Probabilistic Logic Programming'. In: *Inf. Comput.* 101.2 (1992), pages 150–201 (cited on page 102).

[Ono03] Hiroakira Ono. 'Substructural Logics and Residuated Lattices – an Introduction'. In: *Trends in Logic: 50 Years of Studia Logica"*. Edited by Vincent F. Hendricks and Jacek Malinowski. Springer, 2003, pages 193–228 (cited on pages 26, 27).

[OK85] Hiroakira Ono and Yuichi Komori. 'Logics without the contraction rule'. In: *Journal of Symbolic Logic* 50 (1985), 169—201 (cited on page 27).

[Pav79a] Jan Pavelka. 'On Fuzzy Logic I. Many-valued rules of inference'. In: *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik* 25.3-6 (1979), pages 45–52 (cited on page 34).

[Pav79b] Jan Pavelka. 'On Fuzzy Logic II. Enriched residuated lattices and semantics of propositional calculi'. In: *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik* 25 (1979), pages 119–134 (cited on page 27).

[PMW02] Marcus Pearce, David Meredith and Geraint A. Wiggins. 'Motivations and methodologies for automation of the compositional process'. In: *Musicae Scientiae* 6.2 (2002), pages 119–147 (cited on page 59).

[Pnu77] Amir Pnueli. 'The temporal logic of programs'. In: *18th Annual Symposium on Foundations of Computer Science*. IEEE. 1977, pages 46–57 (cited on page 67).

[RG94] Geber Ramalho and Jean-Gabriel Ganascia. 'Simulating Creativity in Jazz Performance'. In: *AAAI 1994*. AAAI Press / The MIT Press, 1994, pages 108–113 (cited on page 80).

[Rei74] Steve Reich. *Writings about Music*. Nova Scotia series. Press of the Nova Scotia College of Art and Design, 1974 (cited on page 94).

[Rei80] Steve Reich. *Clapping music*. Universal Edition, 1980 (cited on pages 70, 94).

[Sac85] Oliver Sacks. *The man who mistook his wife for a hat and other clinical tales*. Summit Books, 1985 (cited on page 9).

[ST12] Donald Sannella and Andrzej Tarlecki. *Foundations of Algebraic Specification and Formal Software Development*. Springer, 2012, pages I–XVI, 1–581 (cited on pages 14, 18, 23, 50, 93).

[San+17] Adam Santoro et al. 'A simple neural network module for relational reasoning'. In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*. 2017, pages 4974–4983 (cited on page 9).

[SFV95] Thomas Schiex, Helene Fargier and Gerard Verfaillie. 'Valued constraint satisfaction problems: Hard and easy problems'. In: *IJCAI (1)* 95 (1995), pages 631–639 (cited on page 45).

[Sch96] Klaus Schmid. 'Making AI systems more creative: the IPC-model'. In: *Knowledge-Based Systems* 9.6 (1996), pages 385 –397 (cited on page 6).

[Sch+14] Marco Schorlemmer et al. 'COINVENT: Towards a Computational Concept Invention Theory'. In: *International Conference on Computational Creativity*. Edited by Simon Colton et al. computationalcreativity.net, 2014, pages 288–296 (cited on page 8).

[Sha83] Ehud Y. Shapiro. 'Logic Programs With Uncertainties: A Tool for Implementing Rule-Based Systems'. In: *Proceedings of the 8th International Joint Conference on Artificial Intelligence. Karlsruhe, FRG, August 1983*. Edited by Alan Bundy. William Kaufmann, 1983, pages 529–532 (cited on page 102).

[Tar86] Andrzej Tarlecki. 'Quasi-varieties in Abstract Algebraic Institutions'. In: *J. Comput. Syst. Sci.* 33.3 (1986), pages 333–360 (cited on page 105).

[Tar95] Andrzej Tarlecki. 'Moving Between Logical Systems'. In: *COMPASS/ADT*. Edited by Magne Haveraaen, Olaf Owe and Ole-Johan Dahl. Volume 1130. LNCS. Springer, 1995, pages 478–502 (cited on page 23).

[TBG91] Andrzej Tarlecki, Rod M. Burstall and Joseph A. Goguen. 'Some Fundamental Algebraic Tools for the Semantics of Computation: Part 3: Indexed Categories'. In: *Theor. Comput. Sci.* 91.2 (1991), pages 239–264 (cited on page 84).

[ŢF15a] Ionuţ Ţuţu and José L. Fiadeiro. 'Revisiting the Institutional Approach to Herbrand's Theorem'. In: *6th Conference on Algebra and Coalgebra in Computer Science, CALCO 2015, June 24-26, 2015, Nijmegen, The Netherlands*. Edited by Lawrence S. Moss and Pawel Sobocinski. Volume 35. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015, pages 304–319 (cited on page 105).

[ŢF15b] Ionuţ Ţuţu and José L. Fiadeiro. 'Service-oriented logic programming'. In: *LMCS* 11.3 (2015), pages 1–38 (cited on pages 82, 103, 113).

[ŢF17] Ionuţ Ţuţu and José L. Fiadeiro. 'From conventional to institution-independent logic programming'. In: *Journal of Logic and Computation* 27.6 (2017), pages 1679–1716 (cited on pages 26, 102).

[Voj01] Peter Vojtás. 'Fuzzy logic programming'. In: *Fuzzy Sets and Systems* 124.3 (2001), pages 361–370 (cited on page 102).

[WD39] Morgan Ward and Robert Palmer Dilworth. 'Residuated Lattices'. In: *Transactions of the American Mathematical Society* 45 (1939), pages 335–354 (cited on page 27).

[Wig06a] Geraint A. Wiggins. 'A preliminary framework for description, analysis and comparison of creative systems'. In: *Knowledge-Based Systems* 19.7 (2006), pages 449–458 (cited on pages 6, 7).

[Wig06b] Geraint A. Wiggins. 'Searching for Computational Creativity'. In: *New Generation Comput.* 24.3 (2006), pages 209–222 (cited on page 7).

[WPM09] Geraint A. Wiggins, Marcus T Pearce and Daniel Müllensiefen. 'Computational modeling of music cognition and musical creativity'. In: *Oxford Handbook of Computer Music*. Oxford University Press, 2009. Chapter 19 (cited on page 59).

[Wir+06] Martin Wirsing et al. 'Semantic-Based Development of Service-Oriented Systems'. In: *FORTE*. Edited by Elie Najm, Jean-François Pradat-Peyre and Véronique Donzeau-Gouge. Volume 4229. LNCS. Springer, 2006, pages 24–45 (cited on page 81).

[Wir+07] Martin Wirsing et al. ‘A Rewriting Logic Framework for Soft Constraints’. In: *Electr. Notes Theor. Comput. Sci.* 176.4 (2007), pages 181–197 (cited on page 81).

INDEX OF NOTATIONS

INDEX OF CONCEPTS