Many-valued Institutions for Constraint Specification

Claudia Elena Chiriţă[1], José Luiz Fiadeiro[1] and Fernando Orejas[2]

[1]Dept. of Computer Science, Royal Holloway University of London, UK
claudia.elena.chirita@gmail.com, jose.fiadeiro@rhul.ac.uk
[2] Dep. de Llenguatges i Sistemes Informàtics, Uni. Politècnica de Catalunya, Spain
orejas@lsi.upc.edu *

Abstract. We advance a general technique for enriching logical systems with soft constraints, making them suitable for specifying complex software systems where parts are put together not just based on how they meet certain functional requirements but also on how they optimise certain constraints. This added expressive power is required, for example, for capturing quality attributes that need to be optimised or, more generally, for formalising what are usually called service-level agreements. More specifically, we show how institutions endowed with a graded semantic consequence can accommodate soft-constraint satisfaction problems. We illustrate our approach by showing how, in the context of service discovery, one can quantify the compatibility of two specifications and thus formalise the selection of the most promising provider of a required resource.

1 Introduction

The problem of supporting the process of building complex systems from simpler parts has deserved a lot of attention since the birth of software engineering, and has been addressed by formal methods of different kinds (e.g. [12]). One such family of formal methods is known under the general heading of 'algebraic specification' (e.g. [23]). In a nutshell, the method is based on the simple principle that parts of software applications (components, modules, and so on) should expose interfaces where they specify required and provided properties. Those parts can then be connected if their interfaces match (in the sense that required properties are met by those provided).

A well-known theory of algebraic specifications is based on the theory of 'institutions' [17]. Essentially, institutions provide logical languages for formulating the properties that will go on the interfaces of parts and an algebra of models that provide mathematical abstractions of the parts; properties and models are related in a way that supports compositionality, i.e. that the properties of a complex whole can be derived from those of its parts.

* Work partially supported by funds from the Spanish Ministry for Economy and Competitiveness (MINECO) and the European Union (FEDER funds) under grant COMMAS (ref. TIN2013-46181-C2-1-R)

The final publication is available at Springer via http://dx.doi.org/10.1007/978-3-662-49665-7_21

That theory is based on exact matches between interfaces, i.e. either the provided properties satisfy the required ones or they do not. Whereas this has served well the specification of functional requirements, software development has evolved in ways that require the specification of properties that can be met in more than one way, i.e. that express 'soft' constraints. A typical example is service-oriented software development where software applications (requesters) can choose among several application suppliers (providers) every time a need for a service arises; the requester first needs to discover a provider that can guarantee, through an interface, the fulfilment of certain requirements, and then to bind to a provider that optimises the satisfaction of certain constraints (e.g. shipment costs in relation to delivery time) establishing a 'service-level agreement'. Another example arises in the context of software product lines, where the selection of features may require the optimisation of given quality attributes of the resulting software variant [18].

In this context, soft-constraint systems have been successfully employed for capturing such non-functional requirements in service-oriented architectures [20, 27], including the negotiation of service-level agreements [6], as well as in the context of software product lines (e.g. [2]). The two main approaches to soft constraint satisfaction problems, SCSP [4] and VCSP [24, 9], generalise the classical crisp variant of constraint satisfaction problems (CSP) by evaluating constraints over c-semirings and valuation structures, respectively.

Our aim in this paper is to extend the institution-based theory of algebraic specifications to address soft constraints. Although the idea of extending abstract data types with soft constraints was already outlined, essentially through examples, in [15], it lacks a rigorous formalisation within the setting of institutions. Such an extension is essential to provide 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, like HETS [21], offer automated support for the specification and analysis of systems.

To this end, in Sec. 2, we first extend the traditional notion of institution along the lines of [10] by replacing the boolean space of truth values with residuated lattices [16], which offer a unifying truth structure for both idempotent c-semirings and valuation structures [3]. Using a simple example, we explain how first-order logic specifications can be extended with soft constraints, and then show how this extension can be generalised to define a logical system of soft constraints as a many-valued institution parameterised by a stratified logic [1]. Based on this construction, in Sec. 3, we formalise the mechanism of selecting a most promising provider of a needed resource in the context of service discovery and binding on the quantification of the compatibility of two constraint specifications as a value of a residuated lattice; we achieve this by defining a compatibility score using the concept of graded semantic consequence [10]. Lastly, in Sec. 4 we study the evolutionary behaviour of service applications. We show how our framework captures situations where different service components (constraint specifications) are based on different truth spaces, which arise in heterogeneous complex systems. We also take into account the dynamicity of preferences during the development

of a system (the change of the truth structures, or of the preferences expressed as sentences of the specifications), and underline the uncertainties of predicting the evolutionary behaviour of service applications. The paper relies on basic knowledge of category theory, for example at the level of [11, 22].

2 Soft-constraint specification in institutions

In this section, after briefly recalling the notion of institution, we focus on the construction of a particular type of institution that is suitable for defining soft CSP specifications. As an example, we describe in more detail how constraint specifications can be written over the institution 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 $\mathrm{CSP}(\mathcal{I})$.

2.1 Institutions

The notion of *institution* was introduced by Goguen and Burstall [17] at the beginning of the 80's to allow for studying concepts for structuring and modularising specifications, independently of the actual formalism to be used for writing the specifications. Intuitively, the notion of institution is an abstract view of the main ingredients of a logical or specification formalism. In particular, an institution consists of:

- A category of signatures, where signatures are the basic elements that we use for building formulas. For instance, in first-order logic, signatures are sets of sorts and function and predicate symbols together with their arity.
- A functor Sen that associates, to each signature Σ, the set of all the formulas that can be written using Σ. In the case of first-order logic, this would mean all the formulas that can be written using the predicate and function symbols in the signature, and including the standard logical connectives and quantifiers. Sen is a functor and not just a mapping, because we want to explicitly associate to each signature morphism $\varphi\colon \Sigma \to \Sigma'$ that translates symbols in Σ into symbols in Σ', the mapping $\mathrm{Sen}(\varphi)$ that translates formulas over Σ into formulas over Σ'.
- A functor Mod that associates, to each signature Σ, the category of all its models. In the case of first-order logic, $\mathrm{Mod}(\Sigma)$ is the category of all Σ-algebras. Again, Mod is a functor and not just a mapping, because we want to explicitly associate to each signature morphism $\varphi\colon \Sigma \to \Sigma'$ that translates symbols in Σ into symbols in Σ', the *reduct* associated to that morphism. In particular, if A' is a Σ'-algebra, its reduct along $\varphi\colon \Sigma \to \Sigma'$ would be a Σ-algebra A, where each symbol s in Σ is interpreted like the symbol $\varphi(s)$ in A'.
- A satisfaction relation that, given a Σ-formula ρ and a Σ-model M, tells us if M satisfies ρ. Moreover, it is required that institutions (i.e. the formalisms

that we consider valid) satisfy the *satisfaction condition* that states that satisfaction does not depend on the choice of signature, i.e. satisfaction is invariant under language translation.

Definition 1 (Institution). *An* institution $\mathcal{I}$ *consists of*

- *a category* $\mathbb{S}\mathrm{ig}^{\mathcal{I}}$ *whose objects are called* signatures*,*
- *a* sentence functor $\mathrm{Sen}^{\mathcal{I}}\colon \mathbb{S}\mathrm{ig}^{\mathcal{I}} \to \mathbb{S}\mathrm{et}$ *giving for every signature* Σ *the set* $\mathrm{Sen}^{\mathcal{I}}(\Sigma)$ *of* Σ-sentences *and for every signature morphism* φ *the* sentence translation map $\mathrm{Sen}^{\mathcal{I}}(\varphi)$*,*
- *a* model functor $\mathrm{Mod}^{\mathcal{I}}\colon (\mathbb{S}\mathrm{ig}^{\mathcal{I}})^{\mathrm{op}} \to \mathbb{C}\mathrm{at}$ *defining for every signature* Σ *the category* $\mathrm{Mod}^{\mathcal{I}}(\Sigma)$ *of* Σ-models *and* Σ-model homomorphisms*, and for every signature morphism* φ *the* reduct functor $\mathrm{Mod}^{\mathcal{I}}(\varphi)$*,*
- *a* satisfaction relation $\models^{\mathcal{I}}_{\Sigma} \subseteq |\mathrm{Mod}^{\mathcal{I}}(\Sigma)| \times \mathrm{Sen}^{\mathcal{I}}(\Sigma)$ *for every signature* Σ*,*

such that the satisfaction condition $\mathrm{Mod}^{\mathcal{I}}(\varphi)(M') \models^{\mathcal{I}}_{\Sigma} \rho \quad \textit{iff} \quad M' \models^{\mathcal{I}}_{\Sigma'} \mathrm{Sen}^{\mathcal{I}}(\varphi)(\rho)$ *holds for any signature morphism* $\varphi\colon \Sigma \to \Sigma'$*,* Σ'*-model* M' *and* Σ*-sentence* ρ*.*

We may omit sub- or super-scripts when there is no risk of confusion. The sentence translation $\mathrm{Sen}^{\mathcal{I}}(\varphi)$ and the reduct functor $\mathrm{Mod}^{\mathcal{I}}(\varphi)$ may also be denoted by $\varphi(\_)$ and $\_{\restriction_\varphi}$. When $M = M'{\restriction_\varphi}$ we say that M is a φ-*reduct* of M' and that M' is a φ-*expansion* of M.

A specification in an institution $\mathcal{I}$ is a pair (Σ, E) consisting of a signature and a collection of sentences (axioms) in the language of that signature, i.e. $E \subseteq \mathrm{Sen}^{\mathcal{I}}(\Sigma)$ – what is usually called a (theory) presentation. A morphism of specifications $\phi\colon (\Sigma, E) \to (\Sigma', E')$ is a signature morphism $\phi\colon \Sigma \to \Sigma'$ such that $E' \models \phi(E)$, i.e. the axioms of (Σ, E) are semantic consequences of (Σ', E') – such a morphism formalises the way (Σ, E) is a part of (Σ', E'). Presentations and their morphisms constitute a category, which we denote by $\mathbb{P}\mathrm{res}^{\mathcal{I}}$.

An example of a specification in first-order logic is given in Fig. 1 (written in a CASL-like syntax [8]) – the specification of residuated lattices, i.e. the first-order structures that satisfy the axioms of the specification are the residuated lattices, which play an essential role in this paper.[1]

2.2 Generalising the truth space

As said above, institutions are an abstraction of logical formalisms, where you describe its main ingredients, in particular, when a given formula is satisfied (or is not satisfied) by a given model. However, when dealing with soft constraints, we need to allow for different degrees of satisfaction. This means, replacing the 'true'/'false' structure of truth values by a more complex kind of structures. In this paper, we consider that these structures are residuated lattices. The choice for residuated lattices is motivated by the fact that the addition of a

[1] The residuated lattices thus specified are sometimes called commutative (because the monoid is commutative), integral (because the unit of the monoid is a greatest element of the lattice), and zero-bounded (because there is a lowest element 0) [16].

spec ResiduatedLattices =
sort Sat
ops 0 : Sat, 1 : Sat
_ ∨ _ : Sat × Sat ⟶ Sat [comm, assoc, unit 0, idem]
_ ∧ _ : Sat × Sat ⟶ Sat [comm, assoc, unit 1, idem]
_ ∗ _ : Sat × Sat ⟶ Sat [comm, assoc, unit 1]
_ → _ : Sat × Sat ⟶ Sat
pred _ ≤ _ : Sat × Sat
∀ a, b, c : Sat
• 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

Fig. 1. The specification $(\Sigma_{\mathbb{RL}}, E_{\mathbb{RL}})$ of residuated lattices

residual operation to semirings and valuation structures has been shown in [3, 7] to provide a unifying framework for soft CSP: residuated lattices generalise both commutative idempotent semirings and fair valuation structures, which are the structures usually employed with local consistency techniques [5].

We actually need for the lattices to be complete (i.e. that a supremum and an infimum exists for every set of degrees of satisfaction).

Definition 2 (Complete residuated lattices). *A* complete residuated lattice $\mathcal{L} = (L, \leq, \vee, \wedge, *, \rightarrow, 0, 1)$ *is a complete lattice (with supremum* $\vee$*, infimum* $\wedge$*, smallest element* 0 *and greatest element* 1*) equipped with a monoidal structure (a commutative and associative binary operation* $*$ *having* 1 *as identity) such that, for all elements* $x, y, z \in L$*,* $(x * y) \leq (x * z)$ *if* $y \leq z$*, and* $y \leq (x \rightarrow z)$ *iff* $x * y \leq z$*.*

A morphism $\lambda\colon \mathcal{L} \to \mathcal{L}'$ *is a function* $\lambda\colon L \to L'$ *that is simultaneously a morphism of complete lattices and of commutative monoids, and is compatible with the residual* $\rightarrow$*. We denote the corresponding category by* $\mathbb{RL}$*.*

Intuitively, the set L provides the degrees of satisfaction (with 0 as dissatisfaction and 1 as total satisfaction) which are ordered according to $\vee$ or, equivalently, to $\wedge$: $a \leq b$ iff $a \vee b = b$. The operation $*$ captures the accumulation of truth values that result from successive inferences, and $\rightarrow$ corresponds to the entailment between two degrees of satisfaction. To capture soft CSP as a many-valued logical system, we therefore extend the notion of institution in keeping with [10]:

Definition 3 ($\mathbb{RL}$-institution). *An* $\mathbb{RL}$-institution $\mathcal{I}$ *is defined as a tuple* $(\mathbb{S}\mathrm{ig}^{\mathcal{I}}, \mathrm{Sen}^{\mathcal{I}}, \mathrm{Mod}^{\mathcal{I}}, \mathcal{RL}^{\mathcal{I}}, \models^{\mathcal{I}})$ *consisting of*

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

such that the equality $\big(\mathrm{Mod}^{\mathcal{I}}(\varphi)(M') \models^{\mathcal{I}}_{\Sigma} \rho\big) = \mathcal{RL}^{\mathcal{I}}(\varphi)\big(M' \models^{\mathcal{I}}_{\Sigma'} \mathrm{Sen}^{\mathcal{I}}(\varphi)(\rho)\big)$ *holds for any signature morphism* $\varphi\colon \Sigma \to \Sigma'$*,* Σ'*-model* M' *and* Σ*-sentence* ρ*. The satisfaction relation extends to a consequence relation over* $E, \Gamma \subseteq \mathrm{Sen}(\Sigma)$ *as follows:* $E \models^{\mathcal{I}}_{\Sigma} \Gamma = \bigwedge\{(M \models^{\mathcal{I}}_{\Sigma} E) \to (M \models^{\mathcal{I}}_{\Sigma} \Gamma) \mid M \in |\mathrm{Mod}(\Sigma)|\}$*.*

The rest of this section is dedicated to showing how, starting from an institution $\mathcal{I}$ that satisfies some structural properties, we can define an $\mathbb{RL}$-institution $\mathrm{CSP}(\mathcal{I})$ of soft-constraint satisfaction problems based on $\mathcal{I}$.

2.3 The first-order soft-constraint $\mathbb{RL}$-institution

To specify systems using constraints, which we evaluate over residuated lattices, we consider only those presentations that extend $(\Sigma_{\mathbb{RL}}, E_{\mathbb{RL}})$, that is presentations (Σ, E) with $\Sigma_{\mathbb{RL}} \subseteq \Sigma$ and $E \models E_{\mathbb{RL}}$. This means that, on the one hand, every (Σ, E)-model has an underlying residuated lattice (its reduct as a $\Sigma_{\mathbb{RL}}$-model) and that, on the other hand, we can make use of the symbols in $\Sigma_{\mathbb{RL}}$ when writing the sentences of E. Moreover, we admit only morphisms of presentations $\varphi\colon (\Sigma, E) \to (\Sigma', E')$ that do not change the symbols of $\Sigma_{\mathbb{RL}}$.

Example 4. Fig. 2 depicts the specification of a customer's book-buying preferences. Customer extends the specification BookData, which 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 return the language in which the book is written and the number of days associated with each delivery mode.

spec BookData = Nat
then sorts Book, Language, Delivery
ops en, fr, de, pt, ro, es : Language
standard, express, online : Delivery
language : Book $\longrightarrow$ Language
deliveryTime : Book $\times$ Delivery $\longrightarrow$ Nat

spec Customer = BookData **and** ResiduatedLattices
then ops languagePref : Language $\longrightarrow$ Sat
deliveryPref : Delivery $\times$ Book $\times$ Nat $\longrightarrow$ Sat
$\forall$ b : Book; n, n' : Nat
• languagePref(en) $\leq$ languagePref(de)
• languagePref(de) $\leq$ languagePref(fr)
• deliveryPref(express, b, n) $\leq$ deliveryPref(online, b, n')
• deliveryPref(standard, b, n) $\leq$ deliveryPref(online, b, n')
• deliveryPref(express, b, n) $\leq$ deliveryPref(standard, b, n') iff n $\geq 3 \wedge$ n' ≤ 7

Fig. 2. The specifications BookData and Customer

Customer also extends the specification of residuated lattices given in Fig. 1 and adds two new function symbols – languagePref and deliveryPref – both of sort Sat. Because every model of Sat is a residuated lattice, the two new function 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))
- deliveryPref(delivery, book, deliveryTime(delivery, book))

A constraint sentence (or constraint for short) is a term of sort Sat. 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 axiomatisation 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* [5].

The extension of first-order logic with constraint sentences is best accommodated in what are called stratified institutions [1], which provide an elegant way of capturing the valuations of constraint variables through *states* of models:

Definition 5 (Stratified institution). *A* stratified institution $\mathcal{I}$ *is defined as a tuple* $(\mathbb{S}\mathrm{ig}^{\mathcal{I}}, \mathrm{Sen}^{\mathcal{I}}, \mathrm{Mod}^{\mathcal{I}}, [\![\_]\!]^{\mathcal{I}}, \models^{\mathcal{I}})$[2] *where*

- $\mathbb{S}\mathrm{ig}^{\mathcal{I}}$, $\mathrm{Sen}^{\mathcal{I}}$ *and* $\mathrm{Mod}^{\mathcal{I}}$ *are as for an institution,*
- $[\![\_]\!]^{\mathcal{I}}$ *is a* stratification*, i.e. a collection of*
  - *functors* $[\![\_]\!]^{\mathcal{I}}_{\Sigma} \colon \mathrm{Mod}^{\mathcal{I}}(\Sigma) \to \mathbb{S}\mathrm{et}$ *for every signature* Σ*, and*
  - *surjective*[3] *natural transformations* $[\![\_]\!]^{\mathcal{I}}_{\phi} \colon [\![\_]\!]^{\mathcal{I}}_{\Sigma'} \Rightarrow \mathrm{Mod}^{\mathcal{I}}(\phi) \,;\, [\![\_]\!]^{\mathcal{I}}_{\Sigma}$ *for every signature morphism* $\phi \colon \Sigma \to \Sigma'$*,*
- *the satisfaction relation* $M \models^{m}_{\Sigma} \rho$ *is parameterised by model states,*

such that, for every $\phi \colon \Sigma \to \Sigma'$, $M' \in |\mathrm{Mod}^{\mathcal{I}}(\Sigma')|$, $m' \in [\![M']\!]^{\mathcal{I}}_{\Sigma'}$, $\rho \in \mathrm{Sen}^{\mathcal{I}}(\Sigma)$*:*
$\mathrm{Mod}^{\mathcal{I}}(\phi)(M') \models^{[\![M']\!]^{\mathcal{I}}_{\phi}(m')}_{\Sigma} \rho$ *iff* $M' \models^{m'}_{\Sigma'} \mathrm{Sen}^{\mathcal{I}}(\phi)(\rho)$.

The stratified version of the institution of first-order logic that we adopt, which will be denoted by FOL, has as signatures pairs $\langle \Sigma, V \rangle$ of a first-order 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 as constants (nullary operation symbols). The models of a signature $\langle \Sigma, V \rangle$ are the Σ-models, while the states

[2] In order to simplify the notation, we will omit the stratified institution in the super-script of the satisfaction relation.

[3] By the surjectivity of the natural transformations we understand that for every morphism $\phi \colon \Sigma \to \Sigma'$ and every $M' \in |\mathrm{Mod}^{\mathcal{I}}(\Sigma')|$, $[\![M']\!]^{\mathcal{I}}_{\phi}$ is surjective.

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 $\chi \in [\![M]\!]_{\langle \Sigma, V\rangle}$ is defined as the satisfaction of ρ in (M, χ), i.e. in the extension of M with the interpretation χ of variables.

Notice that every specification in the institution of first-order logic defines a specification in $\underline{\text{FOL}}$ by choosing an empty set of constraint variables, i.e. we identify a first-order specification such as $(\Sigma_{\mathbb{RL}}, E_{\mathbb{RL}})$ with $(\langle \Sigma_{\mathbb{RL}}, \emptyset\rangle, E_{\mathbb{RL}})$.

We can now summarise the construction of the $\mathbb{RL}$-institution CSP($\underline{\text{FOL}}$) of first-order soft-constraint satisfaction problems:

Signatures. A signature is a pair $(\mathcal{L}, \Delta)$ of a complete residuated lattice $\mathcal{L}$ and an extension $\Delta\colon (\Sigma_{\mathbb{RL}}, E_{\mathbb{RL}}) \to (\langle \Sigma, V\rangle, E)$ of the specification of residuated lattices. We include a residuated lattice as part of a signature in order to let specifiers decide on which space of degrees of satisfaction they want to work with. For simplicity we may denote $(\mathcal{L}, \Delta\colon (\Sigma_{\mathbb{RL}}, E_{\mathbb{RL}}) \to (\langle \Sigma, V\rangle, E))$ by $(\mathcal{L}, \Sigma, V, E)$.

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

Models. The models of $(\mathcal{L}, \Sigma, V, E)$ are the models of $(\langle \Sigma, V\rangle, E)$ whose reducts along Δ are complete and admit a morphism into $\mathcal{L}$. Notice that it would be too restrictive to choose only those models of $(\langle \Sigma, V\rangle, E)$ whose reducts over $\Sigma_{\mathbb{RL}}$ 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 $(\mathcal{L}, \Delta\colon (\Sigma_{\mathbb{RL}}, E_{\mathbb{RL}}) \to (\langle \Sigma, V\rangle, E))$ is a pair (M, f) consisting of a model M of $(\langle \Sigma, V\rangle, E)$ together with a morphism $f\colon M{\restriction_\Delta} \to \mathcal{L}$.

Satisfaction relation. For every constraint signature $(\mathcal{L}, \Sigma, V, E)$ and every model M, we define the value of c over M as the *best level of consistency*:

$$\big((M, f) \models_{(\mathcal{L}, \Sigma, V, E)} c\big) = f\big(\bigvee\nolimits_{\chi \in [\![M]\!]_\Sigma} \mathrm{eval}_{(M,\chi)}(c)\big),$$

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

2.4 The CSP($\mathcal{I}$) $\mathbb{RL}$-institution of soft CSP over $\mathcal{I}$

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

C1. To make residuated lattices available to the specifier, we require the existence of an $\mathcal{I}$-presentation $(\Sigma_{\mathbb{RL}}, E_{\mathbb{RL}})$ such that $\mathbb{RL} \subseteq \mathrm{Mod}^{\mathcal{I}}(\Sigma_{\mathbb{RL}}, E_{\mathbb{RL}})$. This does not restrict applicability as most institutions suitable for the domains where soft constraints are useful will provide the ability to specify data 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 each signature. In addition, we assume that for every object $\Delta\colon (\Sigma_{\mathbb{RL}}, E_{\mathbb{RL}}) \to (\Sigma, E)$ of

the comma category $(\Sigma_{\mathbb{RL}}, E_{\mathbb{RL}})/\mathbb{P}\mathrm{res}^{\mathcal{I}}$ there exists a family of functors $|\_|_{\Sigma}\colon [\![\_]\!]_{\Sigma} \to [\mathrm{C}(\Sigma) \to \mathrm{Mod}^{\mathcal{I}}(\Delta)]$ such that, for any signature morphism $\varphi\colon \Sigma \to \Sigma'$, Σ'-model M', state $\chi' \in [\![M']\!]_{\Sigma'}$, and constraint $c \in \mathrm{C}(\Sigma)$, $|M'{\restriction_\varphi}|_{\Sigma}([\![M']\!]_{\varphi}(\chi'))(c) = |M'|_{\Sigma'}(\chi')(\varphi(c))$.

On this basis, we define the logical system $\mathrm{CSP}(\mathcal{I})$ as follows:

- 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 $(\Sigma_{\mathbb{RL}}, E_{\mathbb{RL}})/\mathbb{P}\mathrm{res}^{\mathcal{I}}$.
- $\mathrm{Sen}^{\mathrm{CSP}(\mathcal{I})}((\mathcal{L}, \Delta\colon (\Sigma_{\mathbb{RL}}, E_{\mathbb{RL}}) \to (\Sigma, E)) = \mathrm{C}(\Sigma)$.
- $\mathrm{Mod}^{\mathrm{CSP}(\mathcal{I})}(\mathcal{L}, \Delta) = \mathrm{Mod}^{\mathcal{I}}(\Delta)/\mathcal{L}$, with $\mathrm{Mod}^{\mathcal{I}}(\Delta)\colon \mathrm{Mod}^{\mathcal{I}}(\Delta)^{-1}(\mathbb{RL}) \to \mathbb{RL}$.
- Given an $(\mathcal{L}, \Delta)$-model $(M, f\colon M{\restriction_\Delta} \to \mathcal{L})$ and a sentence $\rho \in \mathrm{Sen}(\mathcal{L}, \Delta)$, the satisfaction of ρ by (M, f) is defined as:

$$\big((M, f) \models_{(\mathcal{L},\Delta)} \rho\big) = f\big(\textstyle\bigvee_{\chi \in [\![M]\!]_{\Sigma}} |M|_{\Sigma}(\chi)(\rho)\big)$$

Theorem 6. *For any stratified institution $\mathcal{I}$ satisfying the conditions C1 and C2 above,* $\mathrm{CSP}(\mathcal{I})$ *is an $\mathbb{RL}$-institution.*

The following results are important for Section 3.

Proposition 7. $\mathrm{CSP}(\mathcal{I})$ *inherits the following properties of $\mathcal{I}$:*

1. *If $\mathbb{S}\mathrm{ig}^{\mathcal{I}}$ is finitely cocomplete so is $\mathbb{S}\mathrm{ig}^{\mathrm{CSP}(\mathcal{I})}$.*
2. *If $\mathcal{I}$ has (weak) model amalgamation, so does* $\mathrm{CSP}(\mathcal{I})$.
3. *Given factorisation systems [19] $(\mathbb{E}, \mathbb{M})$ for $\mathbb{S}\mathrm{ig}^{\mathcal{I}}$ and $(\mathbb{E}_{\mathbb{RL}}, \mathbb{M}_{\mathbb{RL}})$ for $\mathbb{RL}$, we obtain a factorisation system for $\mathbb{S}\mathrm{ig}^{\mathrm{CSP}(\mathcal{I})}$ by taking the epimorphisms to be the pairs of arrows in $\mathbb{M}_{\mathbb{RL}}$ and $(\Sigma_{\mathbb{RL}}, E_{\mathbb{RL}})/\mathbb{E}^{\mathrm{pres}}$, and the monomorphisms to be the pairs of arrows in $\mathbb{E}_{\mathbb{RL}}$ and $(\Sigma_{\mathbb{RL}}, E_{\mathbb{RL}})/\mathbb{M}^{\mathrm{pres}}$.*

3 Soft constraints for service-oriented computing

As an application of our approach, we study how soft-constraint institutions can be used for formalising structures and processes specific to service-oriented computing: we describe service applications and modules by means of constraint specifications, and define the requirements of applications and the properties guaranteed by service modules as constraint sentences. Consequently, we obtain a series of new results on the way in which service applications evolve through the processes of service discovery, selection, and binding.

We fix an arbitrary $\mathbb{RL}$-institution $(\mathbb{S}\mathrm{ig}^{\mathcal{I}}, \mathrm{Sen}^{\mathcal{I}}, \mathrm{Mod}^{\mathcal{I}}, \mathcal{RL}^{\mathcal{I}}, \models^{\mathcal{I}})$ – see Definition 3 – for which the category of signatures has pushouts, is equipped with a factorisation system, and for which the functor $\mathcal{RL}^{\mathcal{I}}$ preserves pullbacks. In particular, for a soft constraint institution $\mathrm{CSP}(\mathcal{I})$, it suffices that $\mathbb{S}\mathrm{ig}^{\mathcal{I}}$ has pushouts and admits a factorisation system (see Propositions 7.1 and 7.3). We use $\overline{n}$ to denote the set $\{1, \ldots, n\}$.

In our framework of service-oriented computing, for simplicity, we consider that we have two kinds of units, *service applications* and *service modules*. Service

applications can be seen as units that require some services. We may consider that they have an orchestration part, describing what the unit intends to do, and some interfaces describing the services required. In particular, interfaces are subspecifications of the given orchestration together with some property that describes the preferences of the unit to use a given service.

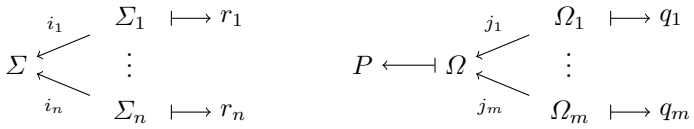
Definition 8 (Service application). *A* service application (Σ, I, R) *consists of a signature* $\Sigma \in |\mathbb{S}\mathrm{ig}|$*, called* orchestration*, together with a finite family* $I = \{i_x\}_{x\in\overline{n}}$ *of* interfaces*, that is, a family of monic signature morphisms* $i_x \colon \Sigma_x \to \Sigma$ *such that* $\mathcal{RL}(\Sigma_x) = \mathcal{RL}(\Sigma)$*, and their associated* requirements $R = \{r_x \in \mathrm{Sen}(\Sigma_x)\}_{x\in\overline{n}}$*. We will refer to a pair* (Σ_x, r_x) *consisting of the domain of an interface and its corresponding requirement as a* requires-specification*.*

Example 9. As part of our running example, we consider a service application $\mathcal{C} = (\Sigma, I, R)$ whose orchestration Σ is Customer (as in Fig. 2), and whose single interface consists of the identity and the requirement

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

Service modules are like service applications but, in addition, they provide functionalities or resources. In this sense, they have an orchestration part and some interfaces for the services required, as well as a provides interface.

Definition 10 (Service module). *A* service module (Ω, P, J, Q) *consists of an* orchestration $\Omega \in |\mathbb{S}\mathrm{ig}|$*, a* provides-property $P \in \mathrm{Sen}(\Omega)$*, a finite family* $J = \{j_y\}_{y\in\overline{m}}$ *of* interfaces $j_y \colon \Omega_y \to \Omega$*, and a family of associated* requirements $Q = \{q_y \in \mathrm{Sen}(\Omega_y)\}_{y\in\overline{m}}$*.*



Example 11. We define a service module $\mathcal{S} = (\Omega, P, J, Q)$ for the application $\mathcal{C}$ given in Ex. 9 by taking Ω as the specification Supplier in Fig. 3, 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 d within $\mathsf{deliveryTime}(b, d)$ days, but in turn it depends on another external delivery-service provider.

Definition 12 (α-satisfiability of an application). *A* service application (Σ, I, R) *is* α-satisfiable *if all of its requirements can be satisfied at once with a value greater than* α*, i.e. there exists a model of its orchestration that satisfies* R *with at least the value* α*:* $\bigvee_{M\in|\mathrm{Mod}(\Sigma)|}\big(\bigwedge_{x\in\overline{n}} M \models i_x(r_x)\big) \geq \alpha$*.*

[4] 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.

spec SUPPLIER = BOOKDATA **and** RESIDUATEDLATTICES
then ops Schiele : Book
available : Book × Delivery ⟶ Sat
deliverable : Book × Delivery × Nat ⟶ Sat
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'
• available(b, d) = 1 ⇔ (b, d) belongs to the following table[4]

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

Fig. 3. The specification SUPPLIER

Definition 13 (β-correctness of a service module). *A* service module $\mathcal{M} = (\Omega, P, J, Q)$ *is said to be* β-correct *if* P *is a consequence of* Q *with a value* $\beta_{\mathcal{M}}$ *greater than* β*. Formally, this means that* $\beta_{\mathcal{M}} = \big(\{j_y(q_y)\}_{y\in\overline{m}} \models_\Omega P\big) \geq \beta$.

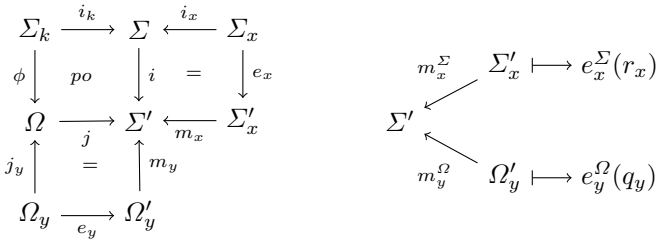
We now focus on the execution of service applications in the context of a fixed set *Rep* of service modules – a *service repository*. Each execution step is triggered by the need to fulfil a requirement of the current application, which in the context of our work corresponds to a requires-specification. 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.

Service discovery. Let $\mathcal{A} = (\Sigma, I, R)$ be a service application and (Σ_k, r_k) 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 – pairs $(\mathcal{M}, \phi)$ of service modules $\mathcal{M} = (\Omega, P, J, Q)$ from *Rep* and *attachment morphisms* ϕ: $\Sigma_k \to \Omega$. Note that the output of the discovery process only depends on the repository and the selected requires-specification, and not on the application itself.

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 $(\mathcal{M}, \phi)$ 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. To this end, we first compute the pushout (i, j) of the signature morphisms i_k and ϕ linking the requires-specification (Σ_k, r_k) 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:

$$\big(j(P) \models \mathrm{Sen}(i_k;i)(r_k)\big) = \bigwedge_{M\in|\mathrm{Mod}(\Sigma')|} \big(M \models_{\Sigma'} j(P)\big) \to \big(M \models_{\Sigma'} \mathrm{Sen}(i_k;i)(r_k)\big).$$

These values belong to different lattices (of different service providers), hence we have to further translate them to the lattice of the service application via the morphisms $\mathcal{RL}(\phi\,;j)$ in order to be able to compare them. Here it is useful to note that $\mathcal{RL}(\phi\,;j) = \mathcal{RL}(\phi)$ because $\mathcal{RL}(j)$ is an identity.



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 $(\mathcal{M}, \phi)$, we have to multiply the score $\mathcal{RL}(\phi)(j(P) \models \mathrm{Sen}(i_k\,;i)(r_k))$ obtained as above with $\beta_{\mathcal{M}}$, the correctness of $\mathcal{M}$. Finally, we will select those service modules for which this product is maximal.

$$sel(Rep, \mathcal{A}, \Sigma_k, r_k) = \underset{(\mathcal{M},\phi)}{\arg\max}\{\beta_{\mathcal{M}} * \mathcal{RL}(\phi)\big(j(P) \models \mathrm{Sen}(i_k\,;i)(r_k)\big)\}$$

Example 14. Consider the repository $Rep = \{\mathcal{S}, \mathcal{S}'\}$ where the new service module $\mathcal{S}' = (\Omega', P', J', Q')$ is such that Ω' is as in Fig. 4, $P' = P$, and $Q' = Q$. When selecting a best supplier for the service application $\mathcal{C}$ from Ex. 9, 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 and OtherSupplier, respectively, 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 language, Book 3 is the least preferred, while 2.1 and 2.2 are the most preferred because languagePref(en) $\leq$ languagePref(de) $\leq$ 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, and thus the selection process chooses $\mathcal{S}'$ as the best supplier.

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* which changes the application as follows:

– The new orchestration is the vertex Σ' of the pushout (i, j).

spec OtherSupplier = BookData **and** ResiduatedLattices
then ops ChagallMaVie, Munch : Book
available : Book × Delivery ⟶ Sat
deliverable : Book × Delivery × Nat ⟶ Sat
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'
• available(b, d) = 1 ⇔ (b, d) belongs to the following table

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

Fig. 4. The specification OtherSupplier

- Apart from the interface i_k corresponding to the distinguished requirement, the interfaces of the application are preserved via a factorisation of the composition of the old interfaces and the morphism of orchestrations i: for $x \in \overline{n} \setminus \{k\}$, we obtain the interface $m_x^{\Sigma} \colon \Sigma'_x \to \Sigma'$ by taking the factorisation $e_x^{\Sigma} \,;\, m_x^{\Sigma}$ of the composed morphism $i_x \,;\, i$.
- The interface i_k is replaced by the interfaces of the selected service module: for $y \in \overline{m}$, $m_y^{\Omega} \colon \Omega'_y \to \Sigma'$ is the monic in the factorisation of $j_y \,;\, j$.
- The distinguished requirement r_k is replaced by the requirements $\{e_y^{\Omega}(q_y)\}_{y \in \overline{m}}$ of the selected module, while the other requirements of the application are kept: for $x \in \overline{n} \setminus \{k\}$, r_x is translated to $e_x^{\Sigma}(r_x)$.

The final goal of binding a service application to different service modules is to obtain an application with all the requirements fulfilled. It is thus natural to be interested in determining a lower bound for the satisfiability of a service application based on the satisfiability of the application that results from the process of binding to a service module with a certain degree of correctness.

Proposition 15 (Correctness of service binding). *Let $\mathcal{M} = (\Omega, P, J, Q)$ be a β-correct module that matches a service application $\mathcal{A} = (\Sigma, I, R)$ 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 $\mathcal{A}' = (\Sigma', I', R')$ of their binding is α-satisfiable, then $\mathcal{A}$ is ζ-satisfiable with $\zeta = \mathcal{RL}(\phi)(\beta * \delta * \alpha)$.*

4 History and value systems

In this section, we analyse two distinguishing features of our method of selecting a best service module: unlike previous boolean approaches [14, 13], it 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 modules will be bound to the application.

4.1 History matters

The choice of a best supplier is usually not invariant to the change of the orchestration of an application. In this section, we identify 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 16. Consider the service application $\mathcal{C}' = (\Sigma', I', R)$ with the orchestration Σ' defined as the specification CUSTOMER of the application $\mathcal{C}$ from Ex. 9 to which we add the sentence

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

and having the same requirement R as $\mathcal{C}$. If we repeat the selection process for $\mathcal{C}'$ and the repository $Rep = \{\mathcal{S}, \mathcal{S}'\}$, 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.

Proposition 17. *Let $\mathcal{A} = (\Sigma, I, R)$ be a service application and (Σ_k, r_k) a requires-specification written over an $\mathbb{RL}$-institution 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, the compatibility score between the requirement r_k of $\mathcal{A}$ and the provides-property of a service module $\mathcal{M} = (\Omega, P, J, Q)$ 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.*

Fact 18. *For a $\mathrm{CSP}(\mathcal{I})$ institution having the model-amalgamation property, a constraint signature morphism $\varphi\colon (\Delta, \mathcal{L}) \to (\Delta', \mathcal{L}')$ in $\mathbb{S}\mathrm{ig}^{\mathrm{CSP}(\mathcal{I})}$, with underlying morphisms $\varphi_{pr}\colon (\Sigma, E) \to (\Sigma', E')$ and $\varphi_{rl}\colon \mathcal{L}' \to \mathcal{L}$, admits model expansions whenever the morphism of presentations φ_{pr} admits model expansions and the reduct $M{\restriction_{\Delta}}$ of any (Σ, E)-model M is projective with respect to φ_{rl}.*

4.2 Changing the truth space

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.

Example 19. Consider once again the service application $\mathcal{C}$ from Ex. 9 and two suppliers $\mathcal{S}_1$ and $\mathcal{S}_2$ whose orchestrations have the same underlying signature – SIMPLESUPPLIER as in Fig. 5. Moreover, they have the same provides-property

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

and no requirements. The residuated lattices of the orchestrations of $\mathcal{S}_1$ and $\mathcal{S}_2$ differ: both $\mathcal{S}_1$ and $\mathcal{C}$ are based on the same Heyting algebra $\mathcal{L}$ with the underlying set of truth values $[0,1]$, while $\mathcal{S}_2$ is based on the real-valued Łukasiewicz lattice $Ł = ([0,1], \min, \max, *, \to, 0, 1)$, with $x * y = \max\{0, x + y - 1\}$ and $x \to y = \min\{1, 1 - x + y\}$, for any $x, y \in [0,1]$. 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 $\mathcal{S}_2$ will be 1 and 0.5, respectively. Consequently, for any match ϕ between $\mathcal{C}$ and $\mathcal{S}_2$ such that the morphism of residuated lattices $\mathcal{RL}(\phi)\colon Ł \to \mathcal{L}$ does not map 0.5 to 1, the selection process will only determine $\mathcal{S}_1$ 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.

spec SIMPLESUPPLIER = BOOKDATA **and** RESIDUATEDLATTICES
then ops available : Book × Delivery ⟶ Sat
deliverable : Book × Delivery × Nat ⟶ Sat

Fig. 5. The specification SIMPLESUPPLIER

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

Example 20. Let $\mathcal{S}_3$ be a service module based on the extension of SIMPLESUPPLIER with the sentence

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

Its provides-property is $P = \mathsf{available}(\mathsf{book}, \mathsf{delivery})$, and it has only one requirement, $\mathsf{deliverable}(\mathsf{book}, \mathsf{delivery}, \mathsf{deliveryTime}(\mathsf{book}, \mathsf{delivery}))$. The correctness of the module $\mathcal{S}_3$ will depend on the residuated lattice of its orchestration: for any Heyting algebra, the module will be correct with the value 1, while for the real-valued Łukasiewicz lattice, the module will only be 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 Conclusions and future work

We have developed a general technique for extending arbitrary institutions with soft constraints that formalises and generalises the results presented in [15]. Our approach consists in adding constraints to specifications written over a base stratified institution that provides functional requirements. The proposed technique requires that the underlying stratified institution $\mathcal{I}$ is expressive enough to capture residuated lattices, which provide the space of degrees of satisfaction in which constraints are expressed, and that every signature of $\mathcal{I}$ provides constraint variables, constraint sentences, and mappings through which each valuation of the constraint variables determines an interpretation of the constraints as elements of the residuated lattice. Building on this formalisation, we have shown how the

selection of a best supplier in the context of service discovery and binding can be defined in terms of graded semantic consequence, and we have studied the unpredictability of the evolution of service applications that originates from the change of the truth structures that underlie the service components.

In order to facilitate an implementation of our model-theoretical approach to choosing a best supplier, we intend to further examine sound and complete proof systems defined in terms of many-valued rules as in [10]. These could be used in the development of operational semantics for the execution of such service-oriented applications (i.e. of a model for dynamic reconfiguration of systems in the style of [14]) with evolving preferences and truth spaces. Towards that end, the logic-programming semantics of services recently proposed in [26] provides a starting point. Besides the obvious need to adapt the theory presented therein to our many-valued setting (which means replacing linear temporal sentences with soft constraint specifications), the main open question is how to generalise the orchestrations of client applications and service modules in order to capture the way in which the satisfaction of constraint sentences changes upon iterations of the processes of service discovery, selection and binding.

We also consider worthwhile investigating how a graded variant of institution-independent logic programming, which generalises service-oriented logic programming, can be defined in relation to the developments presented in [25]. This would necessitate adapting the institution-independent abstractions of the concepts of Herbrand model, unification, resolution and computed answer (with a given degree of confidence) to the many-valued nature of our setting.

Acknowledgements. The authors would like to thank the anonymous referees for their very useful comments and suggestions. These have lead to an improved overall readability of the paper and to a more accurate presentation of the completeness requirement of the residuated lattices.

References

1. M. Aiguier and R. Diaconescu. Stratified institutions and elementary homomorphisms. *Information Processing Letters*, 103(1):5–13, 2007.
2. D. Benavides, P. T. Martín-Arroyo, and A. R. Cortés. Automated reasoning on feature models. In O. Pastor and J. F. e Cunha, editors, *Advanced Information Systems Engineering*, volume 3520 of *LNCS*, pages 491–503. Springer, 2005.
3. S. Bistarelli and F. Gadducci. Enhancing constraints manipulation in semiring-based formalisms. In G. Brewka, S. Coradeschi, A. Perini, and P. Traverso, editors, *ECAI 2006*, volume 141, pages 63–67. IOS Press, 2006.
4. S. Bistarelli, U. Montanari, and F. Rossi. Semiring-based constraint satisfaction and optimization. *Journal of the ACM*, 44(2):201–236, 1997.
5. S. Bistarelli, U. Montanari, F. Rossi, T. Schiex, G. Verfaillie, and H. Fargier. Semiring-based CSPs and valued CSPs: Frameworks, properties, and comparison. *Constraints*, 4(3):199–240, 1999.
6. S. Bistarelli and F. Santini. A nonmonotonic soft concurrent constraint language for SLA negotiation. *Electr. Notes Theor. Comput. Sci.*, 236:147–162, 2009.

7. S. Bova. Soft constraints processing over divisible residuated lattices. In C. Sossai and G. Chemello, editors, *Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, volume 5590 of *LNCS*, pages 887–898. Springer, 2009.
8. CoFI (The Common Framework Initiative). CASL *Reference Manual*, volume 2960 of *LNCS*. Springer, 2004.
9. D. A. Cohen, M. C. Cooper, P. Jeavons, and A. A. Krokhin. Soft constraints: Complexity and multimorphisms. In F. Rossi, editor, *Principles and Practice of Constraint Programming*, volume 2833 of *LNCS*, pages 244–258. Springer, 2003.
10. R. Diaconescu. Graded consequence: an institution theoretic study. *Soft Comput.*, 18(7):1247–1267, 2014.
11. J. L. Fiadeiro. *Categories for software engineering*. Springer, 2005.
12. J. L. Fiadeiro. The many faces of complexity in software design. In M. Hinchey and L. Coyle, editors, *Conquering Complexity*, pages 3–47. Springer, 2012.
13. J. L. Fiadeiro and A. Lopes. An interface theory for service-oriented design. *Theor. Comput. Sci.*, 503:1–30, 2013.
14. J. L. Fiadeiro and A. Lopes. A model for dynamic reconfiguration in service-oriented architectures. *Software and System Modeling*, 12(2):349–367, 2013.
15. J. L. Fiadeiro and F. Orejas. Abstract constraint data types. In R. D. Nicola and R. Hennicker, editors, *Software, Services, and Systems*, volume 8950 of *LNCS*, pages 155–170. Springer, 2015.
16. N. Galatos, P. Jipsen, T. Kowalski, and H. Ono. *Residuated Lattices: An Algebraic Glimpse at Substructural Logics*. Studies in Logic and the Foundations of Mathematics. Elsevier Science, 2007.
17. J. A. Goguen and R. M. Burstall. Institutions: Abstract model theory for specification and programming. *Journal of the ACM*, 39(1):95–146, 1992.
18. M. Harman, Y. Jia, J. Krinke, W. B. Langdon, J. Petke, and Y. Zhang. Search based software engineering for software product line engineering: a survey and directions for future work. In S. Gnesi, A. Fantechi, P. Heymans, J. Rubin, K. Czarnecki, and D. Dhungana, editors, *Software Product Line*, pages 5–18. ACM, 2014.
19. H. Herrlich and G. Strecker. *Category theory: an introduction*. Allyn and Bacon series in advanced mathematics. Allyn and Bacon, 1973.
20. M. M. Hölzl, M. Meier, and M. Wirsing. Which soft constraints do you prefer? *Electr. Notes Theor. Comput. Sci.*, 238(3):189–205, 2009.
21. T. Mossakowski, C. Maeder, and K. Lüttich. The heterogeneous tool set, HETS. In O. Grumberg and M. Huth, editors, *TACAS*, volume 4424 of *LNCS*, pages 519–522. Springer, 2007.
22. B. C. Pierce. *Basic category theory for computer scientists*. Foundations of computing. MIT Press, 1991.
23. D. Sannella and A. Tarlecki. *Foundations of Algebraic Specification and Formal Software Development*. Springer, 2012.
24. T. Schiex, H. Fargier, and G. Verfaillie. Valued constraint satisfaction problems: Hard and easy problems. *IJCAI (1)*, 95:631–639, 1995.
25. I. Ţuţu and J. L. Fiadeiro. From conventional to institution-independent logic programming. *Journal of Logic and Computation*, 2015.
26. I. Ţuţu and J. L. Fiadeiro. Service-oriented logic programming. *Logical Methods in Computer Science*, 11(3):1–38, 2015.
27. M. Wirsing, A. Clark, S. Gilmore, M. M. Hölzl, A. Knapp, N. Koch, and A. Schroeder. Semantic-based development of service-oriented systems. In E. Najm, J. Pradat-Peyre, and V. Donzeau-Gouge, editors, *FORTE*, volume 4229 of *LNCS*, pages 24–45. Springer, 2006.