A fuzzy approach for negotiating quality of services

Davide Bacciu, Alessio Botta, and Hernán Melgratti

IMT Lucca Institute for Advanced Studies, Italy.
{davide.bacciu, alessio.botta, hernan.melgratti}@imtlucca.it

Abstract. A central point when integrating services concerns to the description, agreement and enforcement of the quality aspect of service interaction, usually known as Service Level Agreement (SLA). This paper presents a framework for SLA negotiation based on fuzzy sets. We propose (i) a request language for clients to describe quality preferences, (ii) a publication language for providers to define the qualities of their offered services, and (iii) a decision procedure for granting any client request with a SLA contract fitting the requestor requirements. We start with a restricted framework in which the different qualities of a service are handled independently (as being orthogonal) and then we propose an extension that allows clients and providers to express dependencies among different qualities.

1 Introduction

Service Level Agreement (SLA) concerns the description, negotiation and enforcement of non-functional aspects of service behaviors. Typical examples are the bounds on service response time and availability, the number of accepted requests by unit of time, and the availability of resources such as storage and bandwidth. In this context, service providers advertise their functionalities by associating different *service levels* or guaranties about the qualities of their offered services. For instance, a storage service may be published with, e.g., three different service levels, namely, *Basic*, *Gold*, and *Platinum*, associating to any of them an increasing amount of communication bandwidth. In this kind of scenario, a client (or service consumer) $\mathcal{C}$ is able to use (or interact with) a particular service provider $\mathcal{P}$ only after $\mathcal{C}$ and $\mathcal{P}$ have agreed on a particular service level. Therefore, any interaction of $\mathcal{C}$ with $\mathcal{P}$ should be preceded by a *negotiation* or *agreement creation* phase. The negotiation phase is started when the client makes a specific request to the provider containing its quality expectations. After a negotiation, if they reach an agreement, a particular contract binding the provider and the client with a particular service level is signed by both parties. After that, a second phase is started: the *utilization* phase. During the utilization phase, the client makes requests to the provider under a particular agreement or signed contract. It is assumed that the service provider will perform a service accordingly to the agreed conditions. Hence, the runtime infrastructure should provide ways for monitoring and checking whether the execution meets contract obligations, and to take corrective actions when the execution deviates from the agreed conditions.

The web services realm have gave birth to some proposals for standards, like *Web Service Level Agreement* (WSLA) [6] and *Web Service Agreement Specification* (WS-Agreement) [13], that specify the way in which services describe their quality levels (or

SLA parameters), the protocols to be used for reaching agreements, how contracts are written, and how systems are monitored. As usual for these kind of specifications, they are long documents exposing the XML syntax of definitions and informally defining the semantics of the different constructs. This paper is aimed at proposing a formal approach for negotiating SLA based on *fuzzy set theory*.

Fuzzy set theory and *fuzzy logic* were introduced by Zadeh [14] to provide mathematical tools that could deal with the vagueness and the uncertainty that are typical of the human perception and reasoning process. The basic idea of the fuzzy approach is to allow an element s to belong to a set φ with degrees of membership ranging in the continuous real interval $[0,1]$, rather than in $\{0,1\}$. The use of fuzzy descriptions intuitively corresponds to the vagueness that can be always found in service level requests. For instance, suppose that $\mathcal{C}$ is looking for a FTP service with a bandwidth of "***about** 5 Mbps*": what happens if the offered bandwidth is just 4.9 Mbps? If the constraint is expressed in a standard crisp way such as $Bandwidth \geq 5\ Mbps$ (i.e. traditional non-fuzzy approach), this *almost acceptable* solution would be immediately discarded, while it could turn out that actually this is the *best* solution from a cost-similarity trade-off viewpoint. Placing a lower threshold on the minimum required bandwidth (e.g. 4.5 Mbps) does not solve the problem, since an *almost acceptable* solution could always be found (trivially, 4.4 Mbps) and since this constraint does not discriminate very good solutions like 5 Mbps from just acceptable solutions like 4.6 Mbps.

In this paper we focus on the contract creation phase, in which

1. service providers should be able to publish a description of the quality levels they may provide,
2. clients should be able to precisely describe their quality requirements,
3. a decision procedure allows clients and providers to reach agreements.

In particular, we propose a SLA framework based on fuzzy sets for supporting those activities. The proposed framework is presented by describing the following four elements:

- The *negotiation language*, called the SLA-calculus, that models the negotiation phase of the SLA and accounts for the creation and revocation of contracts. SLA-calculus is parametric with respect to the languages used for publishing services and making requests, and to the decision procedure used for reaching an agreement. Its main scope is to identify the key concepts involved in the negotiation phase.
- The *publication language* for specifying the offered service levels.
- The *request language* for specifying the quality levels desired by the user.
- The *agreement procedure* that allows a client $\mathcal{C}$ and service provider $\mathcal{P}$ to reach an agreement.

We remark that the negotiation language is orthogonal to the remaining elements. In particular, we provide two different instantiations of the framework. Fistly, we propose two basic languages for publications and requests in which all SLA parameters are handled independently. Then, we extend those languages (and consequently the agreement procedure) by allowing the definition of dependencies among different SLA parameters (for instance to say that the storage service never provide a low bandwidth and a large disk space).

Related works. Several approaches in the literature [7,11,5] have used the theory of fuzzy sets for studying the problem of finding suitable compositions of services (i.e., discovering appropriate services) that meet certain user-defined QoS requirements. We remark that our approach has a different aim, since it provides a mechanism that allows two specific services (i.e., the client and the provide) to negotiate a particular QoS level — a phase that takes place after the candidate provider has been identified. The work in [10] presents a process calculus accounting for QoS. The main idea is that any interaction is enriched with a constraint that describes its associated QoS. We envisage this approach as an appropriate model for the utilization phase, i.e., once the agreement has been reached. A calculus for dealing with negotiations, called cc-pi, is introduced in [2]. Differently from our approach, a client request and the offered service levels are described in cc-pi as constraints. There is an agreement when both constraints are consistent.

Paper Organization Section 2 presents an overview of fuzzy set theory. Section 3 introduces the negotiation language, while Sections 4 and 5 describe two different instantiations for the publication and request languages and the corresponding decision procedure.

2 Background

This Section summarizes the basics of the fuzzy set theory. A fuzzy set is defined as

Definition 2.1 (Fuzzy set [14]). *Given a space of objects S ranged over by s, a fuzzy set φ in S is characterized by a membership function $\mu_\varphi(s) : S \to [0,1] \subset \mathbb{R}$. $\mu_\varphi(s)$ is the fuzzy degree of membership to which a generic element s belongs to φ.*

Given the fuzzy sets φ, ϕ, and ω defined on the same universe S, the usual concepts of set theory can be generalized by the following definitions.

Definition 2.2 (Empty fuzzy set). $\varphi = \varnothing \Leftrightarrow \forall s \in S, \mu_\varphi(s) = 0$.

Definition 2.3 (Equality). $\varphi = \phi \Leftrightarrow \forall s \in S, \mu_\varphi(s) = \mu_\phi(s)$.

Definition 2.4 (Intersection). *Let* $\omega = \varphi \cap \phi$, $\mu_\omega(s) = \min\{\mu_\varphi(s), \mu_\phi(s)\}$.

Definition 2.5 (Union). *Let* $\omega = \varphi \cup \phi$, $\mu_\omega(x) = \max\{\mu_\varphi(s), \mu_\phi(s)\}$.

The *similarity operator* $\mathrm{Sim}(\cdot,\cdot)$ measures the degree to which two fuzzy sets are equal. This operator is more powerful than the binary equality in Def. 2.3, and it will be used in the following sections to compare fuzzy sets. Intuitively, the operator $\mathrm{Sim}(\cdot,\cdot)$ is defined such that $\mathrm{Sim}(\varphi,\phi) = 0 \Leftrightarrow \varphi \cap \phi = \varnothing$ and $\mathrm{Sim}(\varphi,\phi) = 1 \Leftrightarrow \varphi = \phi$. Intermediate values in $[0,1]$ should be associated to gradually overlapping fuzzy sets. We choose the following definition for $\mathrm{Sim}(\cdot,\cdot)$:

$$\mathrm{Sim}(\varphi,\phi) = \frac{\sum_{s \in S} \mu_{\varphi \cap \phi}(s)}{\sum_{s \in S} \mu_{\varphi \cup \phi}(s)} \ . \tag{1}$$

Note that many similarity operators are available in the literature [12]. Likewise, there is no unique definition for intersection and union in fuzzy set theory: for the sake of simplicity, we select the most popular and simple implementation, but actually any *t-norm* $\otimes$ could be used as intersection and any *t-conorm* $\oplus$ could be used as union [3].

An important property of fuzzy sets is the possibility of extending traditional crisp functions to work on fuzzy sets, via the *extension principle* [3].

Definition 2.6 (Extension principle). *Let $f(s) : S \to T$ be a crisp function and φ a fuzzy set in S. Then, $\psi = f(\varphi)$ is a fuzzy set in T such that $\forall t \in T, \mu_\psi(t) = \sup_{s|t=f(s)} \mu_\varphi(s)$.*

Conversely, sometimes we may want to "defuzzify" fuzzy sets and to extract a crisp value that could serve as a prototype of the whole fuzzy set in crisp applications. To this aim, we need one of the many defuzzification methods that can be found it the literature. We will use the well known *center of gravity* method (defined as in [1]), which is particularly suitable for managing sets of numerical elements

$$\mathrm{cog}(\varphi) = \frac{\sum_{s \in S} s \cdot \mu_\varphi(s)}{\sum_{s \in S} \mu_\varphi(s)} \ . \tag{2}$$

In the rest of the paper we will use the following notion of *linguistic variable*.

Definition 2.7 (Linguistic variable). *A linguistic variable is a variable whose values are linguistic terms. A linguistic variable V is a quintuple $(x, \mathbb{T}, S, G, \mathcal{M})$, where:*

1. *x is the name of the variable (e.g.,* Age*);*
2. *$\mathbb{T}$ is the set of linguistic terms of variable x (e.g.,* {young, old}*);*
3. *S is the universe of discourse of the base variable;*
4. *G is a syntactic rule for generating composed terms of x (e.g., "*very old*");*
5. *$\mathcal{M} : F(S) \to \mathbb{T}$ is a semantic rule mapping a fuzzy set φ defined over the universe of discourse S of the base variable with each element $t \in \mathbb{T}$.*

Even thought we will not use the syntactic rule *G* in the following, we included it in Def. 2.7 for the sake of completeness. Note that this definition of linguistic variable is slightly different from the ones that can be found in the literature [15], which actually defines $\mathcal{M} : \mathbb{T} \to F(S)$. In fact, as it will be clear in Section 5.1, in some special cases we may want to associate a linguistic term with more than one fuzzy set. This anyway does not change the essence of $\mathcal{M}$, which is to capture the semantic mapping between linguistic terms and fuzzy sets.

3 SLA-calculus

In this Section we provide an operational model for the agreement phase, which is parametric with respect to the languages used for describing services and for making quality requests. Its main scope is to expose the ingredients of the model, by showing

how contracts are created, used and then revoked. A main assumption of this model is that the quality levels offered by a provider do not depend on the provider's internal state or, in other words, on the contracts it has already signed. For instance, the bandwidth offered by the storage server do not decrease as new contracts are signed. Moreover, we assume that providers do not revoke contracts. The following are the main entities of the model:

- *Service descriptions*, any of them declares the SLA parameters offered by a particular provider. Any service description may be thought as an entry on a UDDI registry. We rely on an infinite set $\mathcal{S}$ of service names ranged over by $s, s_0, \ldots$. Moreover, we assume SLA parameters to be described by a *provider descriptor* $D_{\mathcal{P}}$, which is a valid document of the publication language.
- *Service states*, we associate any service with a state. In our basic version, a service state collects the information of all active contracts signed by the provider. We assume $\mathbb{Q}$ to be the infinite set of contract names ranged over by $c, c_0, \ldots$.
- *Clients* describe the behavior of applications attempting to sign and revoke contracts with providers. As explained before, a client initiates a negotiation with a provider by sending a *request descriptor* D_C specifying the desired qualities of the service. The request descriptor is any valid document of the request language. If the requested SLA parameters can be assured by the provider, then the client will obtain a signed contract. Otherwise, the negotiation phase will fail.

Syntax The following grammar defines the terms of the SLA-calculus:

$$\begin{array}{ll} (\textsc{Net}) & N ::= s[D_{\mathcal{P}}] \mid s\{c_0, \ldots, c_n\} \mid C \mid N|N \mid (\nu c)N \\ (\textsc{Client}) & C ::= 0 \mid c := s\langle D_C, A\rangle ?C : C \mid \dagger c.C \mid C|C \end{array}$$

A net (system) is either a service description $s[D_{\mathcal{P}}]$, where: (i) s is the service name and $D_{\mathcal{P}}$ is a description of the SLA parameters (the definition of provider descriptors are in the following Sections); (ii) a service state $s\{c_0, \ldots, c_n\}$, denoting that the provider s has signed the active contracts $c_0, \ldots, c_n$; (iii) a client C, (iv) the parallel composition of nets, and (iv) the declaration $(\nu c)N$ of a fresh contract c to be used in N.

A client is either (i) the inert process 0; (ii) a process $c := s\langle D_C, A\rangle ?C_1 : C_2$ that attempts to create a new contract c by negotiating with s for the qualities described by D_C and accepting under conditions A, if the negotiation succeeds then C_1 is executed (otherwise C_2 is activated); (iii) a process $\dagger c.C$ that revokes a signed contract c and then behaves like C; and (iv) the parallel composition $C_1|C_2$.

The only bound names are the occurrences of c in either $c := s\langle D, A\rangle ?C_1 : C_2$ or in $(\nu c)N$. All other occurrences are considered free. We will refer to terms up-to α-equivalence (denoted by $\equiv_\alpha$), i.e., up-to the renaming of bound names.

Operational Semantics The reduction semantics of SLA-calculus is given up-to structural equivalence given by the the rules defining '|' as an associative and commutative operator with 0 as identity and the following ones.

$$\begin{array}{lll} s\{c\} \mid s\{c_1, \ldots, c_n\} \equiv s\{c, c_1, \ldots, c_n\} & N_1 \equiv N_2 & \textit{if } N_1 \equiv_\alpha N_2 \\ (\nu c_1)(\nu c_2)N \equiv (\nu c_2)(\nu c_1)N & N_1|(\nu c)N_2 \equiv (\nu c)(N_1|N_2) & \textit{if } c \notin fn(N_1) \end{array}$$

The reduction semantics is inductively defined by the following rules.

(AGREEMENT)
$$\frac{D_{\mathcal{P}} \approx_{\mathcal{A}} D_C}{s[D_{\mathcal{P}}] \mid c := s\langle D_C, A\rangle ?C_1 : C_2 \;\rightarrow\; s[D_{\mathcal{P}}] \mid (\nu c)(C_1 \mid s\{c\})}$$

(PAR)
$$\frac{N_1 \rightarrow N_1'}{N_1 \mid N_2 \rightarrow N_1' \mid N_2}$$

(DISAGREEMENT)
$$\frac{D_{\mathcal{P}} \not\approx_{\mathcal{A}} D_C}{s[D_{\mathcal{P}}] \mid c := s\langle D_C, A\rangle ?C_1 : C_2 \;\rightarrow\; s[D_{\mathcal{P}}] \mid C_2}$$

(RESTRICTION)
$$\frac{N_1 \rightarrow N_1'}{(\nu c)N_1 \rightarrow (\nu c)N_1'}$$

(REVOKE) $\quad (\nu c)(\dagger c.C \mid s\{c\}) \;\rightarrow\; C$

Rule AGREEMENT stands for the creation of a contract. In this case there is a client $c := s\langle D_C, A\rangle ?C_1 : C_2$ starting a negotiation with the provider s by requiring the service level D_C and accepting the negotiation under conditions A. Since the levels $D_{\mathcal{P}}$ offered by the provider satisfy the user requirement (premise $D_{\mathcal{P}} \approx_{\mathcal{A}} D_C$), a new contract c (known only by the provider and the client) is created. Note this rule abstracts away from the actual agreed conditions, they are represented just by a contract name. Moreover, the service description $s[D_{\mathcal{P}}]$ is persistent. Rule DISAGREEMENT handles the cases in which the provided qualities $D_{\mathcal{P}}$ do not match the client requirements D_C and A. In this cases the exceptional flow C_2 is activated. Rule REVOKE handles the termination of a contract by decision of a client. Rules PAR and RESTRICTION are the standard ones.

Note the rules are parametric with respect to the relation $D_{\mathcal{P}} \approx_{\mathcal{A}} D_C$, which stands for the decision procedure. Examples of such relation are in the following Sections.

4 The Fuzzy Agreement Process (FAP)

This Section instantiates the SLA-calculus presented in the previous Section by proposing specific publication and requirement languages and a decision procedure. We called this particular instance the FAP. The aim of the FAP is to mimic the complex SLA interactions performed by humans, by means of a process in which (i) tolerance and vagueness are admitted both in request and offer specifications, and (ii) the matching of request and offers is evaluated with respect to the trade-offs between similarity metrics and cost considerations.

The following example illustrates the main ingredients of FAP. Let us assume a provider $\mathcal{P}$ offering a FTP service, which is described by a set of K qualities or resources. Any quality x_i is associated with k_i different service levels. Each service level has an associated cost, which is determined by a secret policy of the provider.

Example 4.1. The provider $\mathcal{P}$ offers a FTP service, characterized by the resources $x_1 =$ *Storage* and $x_2 =$ *Bandwidth*, any of them offered with three different service levels, namely $\{Basic, Gold, Platinum\}$ for *Storage* ($k_1 = 3$) and $\{Slow, Medium, Fast\}$ for *Bandwidth* ($k_2 = 3$).

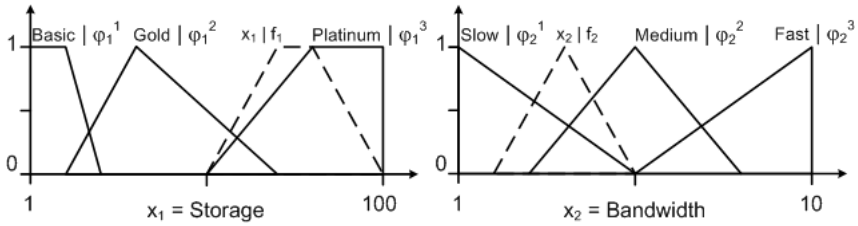

Fig. 1. Linguistic variables and fuzzy sets for the FTP service example.

A *service configuration* associates a service level to any resource of the service. For the FTP example, $PlatinumStorage \wedge FastBandwidth$ is a valid service configuration for our FTP service. In our approach, each resource is modeled by a linguistic variable, while service levels and expected qualities are expressed by using fuzzy sets defined on the universes of discourse of those linguistic variables.

Example 4.2. Fig. 1 shows the fuzzy sets describing the service levels offered by $\mathcal{P}$ (solid lines) and those corresponding to the requirements of a client $\mathcal{C}$ (dotted lines). All fuzzy sets are defined on the universes of discourse of the corresponding linguistic variables *Storage* and *Bandwidth*.

The following definitions formalize the publication language (i.e., the provider descriptors), the request language (i.e., the client descriptors) and the decision procedure (i.e., the relation $\approx_{\mathcal{A}}$) for the FAP instantiation.

Definition 4.1 (Provider descriptor). *A provider descriptor $D_{\mathcal{P}}$ is a pair $(\mathbb{V},\mathbb{C})$. $\mathbb{V}$ is a set of K linguistic variables $V_1,...,V_K$. $\mathbb{C}$ is a set of K cost functions $\Downarrow_1,...,\Downarrow_K$, such that $\Downarrow_i$: $S_i \rightarrow U$, where S_i is the universe of discourse of the* i*-th linguistic variable V_i and U is the target cost universe.*

As aforementioned, linguistic variables describe resources. The name x_i of the linguistic variable V_i gives the name of the resource. We write $\Sigma = \{x_i,...,x_K\}$ for the set of names of all linguistic variables. The service levels associated to the resource x_i are the linguistic terms $\mathbb{T}_i$ of V_i, which are associated by $\mathcal{M}_i$ to the fuzzy sets $\varphi_i^1,...,\varphi_i^{k_i}$ defined on the corresponding universe of discourse S_i (as stated in Def. 2.7).

Example 4.3. In our FTP service, we define service levels $\mathbb{T}_1 = \{Basic, Gold, Platinum\}$ and $\mathbb{T}_2 = \{Slow, Medium, Fast\}$. Some examples of set-label associations are $\mathcal{M}_1(\varphi_1^3) = Platinum$ and $\mathcal{M}_2(\varphi_2^1) = Slow$.

Cost functions $\Downarrow_1,...,\Downarrow_K$ are used to map each resource into a common reference universe U, and they will be used during the decision procedure to compute a global cost of a service configuration. Cost can be expressed in terms of money, time, availability or any other business-related meaningful measure. Different shapes of cost functions can represent different cost models: for instance, an important resource may have a steeper cost function than a less critical resource.

Definition 4.2 (Client descriptor). *A client descriptor D_C is a pair $(\mathbb{F}, \mathcal{A})$. $\mathbb{F}$ is a set of K pairs $\{(x_1|f_1), ..., (x_K|f_K)\}$, where f_i is a fuzzy set defined on the universe of discourse V_i of the linguistic variable named $x_i \in \Sigma$. $\mathcal{A}$ is an acceptance function $U \times [0,1] \rightarrow \{0,1\}$.*

$\mathbb{F}$ describes the configuration desired by the client. Indeed, a pair $(x_i|f_i)$ states that the client expects the service level described by fuzzy set f_i for the resource x_i. In this way a client $\mathcal{C}$ may formulate a vague request such as *"I want a bandwidth of **about** 5 Mbps"*. Note that, since fuzzy set theory extends classic set theory, it is still possible to express precise and mandatory constraints such as *"I want a bandwidth of **at least** 2 Mbps"* or *"I want a bandwidth of **exactly** 4 Mbps"*.

The acceptance function $\mathcal{A}$ is used as a classifier to discriminate acceptable from unacceptable service configurations offered by $\mathcal{P}$. In other words, $\mathcal{A}$ measures the trade-off between the cost and the similarity of the proposed configuration with respect to the original service request and decides if that solution is acceptable or not.

We remark that a client only needs the following information to be able to build a request descriptor:

1. the resource names $\Sigma = \{x_1, ..., x_K\}$;
2. the universes of linguistic variable $\{S_1, ..., S_K\}$;
3. the cost reference universe U.

In a real implementation, this information should be exposed by $\mathcal{P}$ in its service description, e.g., in its corresponding WSDL description. We remark that a client $\mathcal{C}$ needs no information about available service levels nor cost policies used by the provider. Hence, they can be kept private by the provider.

Finally, we define the decision procedure of FAP. Roughly, we say $D_C \approx_{\mathcal{A}} D_{\mathcal{P}}$ if and only if there exists (at least) a service configuration on which the client and the provider agree.

Definition 4.3 (Evaluation of $\approx_{\mathcal{A}}$). *Given $D_C = (\mathbb{F}, \mathcal{A})$ and $D_{\mathcal{P}} = (\mathbb{V}, \mathbb{C})$:*

Step 0. *For each resource x_i $(i = 1...K)$, and for each service level φ_i^j of x_i $(j = 1...k_i)$, let $\phi_i^j = \Downarrow_i (f_i \cap \varphi_i^j)$ be the projection on U of the overlapping between the offered level φ_i^j and the requested level f_i. Furthermore, let $s_i^j = \mathrm{Sim}(f_i, \varphi_i^j)$ be the similarity measure of the same couple of fuzzy sets.*

Step 1. *Let $\Pi = \{\pi | \pi = (\phi_1^{j_1}, ..., \phi_K^{jK}), \forall i = 1...K, \forall j_i = 1...k_i \wedge s_i^{j_i} \neq 0\}$ be the set of all K-tuples representing eligible service configurations offered by $\mathcal{P}$ that have non-empty intersection with the request descriptor. Then, $\forall \pi \in \Pi$, let $\sigma(\pi) = \mathrm{cog}(\bigcup_{i=1}^{K} \phi_i^{j_i})$ and $\mathrm{Sim}_\pi(\pi) = \prod_{i=1}^{K} s_i^{j_i}$ be respectively the crisp global cost and the global similarity measure of π.*

Step 2. *Let $\Pi' = \{\pi \in \Pi | \mathcal{A}(\sigma(\pi), \mathrm{Sim}_\pi(\pi)) = 1\}$ be the set of eligible service configurations accepted by $\mathcal{C}$.*

Step 3.a. *If $\Pi' \neq \varnothing$ then $D_C \approx_{\mathcal{A}} D_{\mathcal{P}}$ and let $\pi^* = \mathrm{choose}(\Pi')$ be the service configuration selected by $\mathcal{P}$ś choice function.*

Step 3.b. *If $\Pi' = \varnothing$ then $\neg(D_C \approx_{\mathcal{A}} D_{\mathcal{P}})$.*

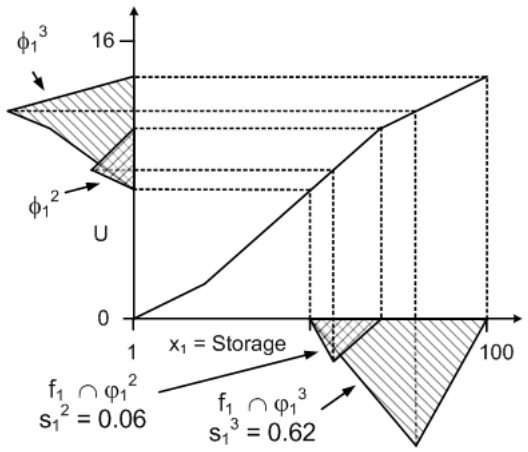

Fig. 2. The cost function of resource *Storage* and a sample application.

Note that we can pass from the fuzzy sets to the linguistic description of each π at any time by simply applying $\bigwedge_{i=1}^{K} \mathcal{M}_i(\varphi_i^{j_i})$.

Example 4.4. We compute $\approx_{\mathcal{A}}$ for the descriptors in Fig. 1. As stated in Section 2, we use the min function as t-norm $\otimes$ and the max function as t-cornorm $\oplus$ in the implementation of $\cap$ and $\cup$, respectively (see Defs. 2.4 and 2.5).

Step 0. Fig. 2 shows the application of **Step 0** to resource *Storage*. First, fuzzy sets resulting from $f_1 \cap \varphi_1^1$, $f_1 \cap \varphi_1^2$ and $f_1 \cap \varphi_1^3$ are obtained in the original universe of discourse of *Storage* and their similarity measure is computed. Note that, as we could expect by observing Fig. 1, $f_1 \cap \varphi_1^1 = \varnothing$ and thus $s(f_1, \varphi_1^1) = 0$, while $s(f_1, \varphi_1^3) \gg s(f_1, \varphi_1^2) > 0$. Then, we use the cost function $\Downarrow_1$ and the extension principle of Def. 2.6 to project the intersections from the universe S_1 to U and thus to obtain ϕ_1^2 and ϕ_1^3 (since φ_1^1 is empty, ϕ_1^1 is also empty). The same procedure is repeated for *Bandwidth* using a very steep linear cost function $\Downarrow_2$ (i.e., *Bandwidth* is a very costly resource), obtaining $s(f_2, \varphi_2^1) = 0.34$, $s(f_2, \varphi_2^2) = 0.22$ and $s(f_2, \varphi_2^3) = 0$.

Step 1. Since $s(f_1, \varphi_1^1) = s(f_2, \varphi_2^3) = 0$, we have four eligible service configurations to include in Π: $\pi_a = (\varphi_1^2, \varphi_2^1), \pi_b = (\varphi_1^2, \varphi_2^2), \pi_c = (\varphi_1^3, \varphi_2^1)$ and $\pi_d = (\varphi_1^3, \varphi_2^2)$, which correspond to the linguistic descriptions *GoldStorage* $\wedge$ *SlowBandwidth*, *GoldStorage* $\wedge$ *MediumBandwidth*, *PlatinumStorage* $\wedge$ *SlowBandwidth* and *PlatinumStorage* $\wedge$ *MediumBandwidth*, respectively. We compute $\sigma(\pi_a) = \mathrm{cog}(\varphi_1^2 \cup \varphi_2^1) = 9.6$ and $\mathrm{Sim}_\pi(\pi_a) = s_1^2 \cdot s_2^1 = 0.06 \cdot 0.34 = 0.0204$, and similarly for the remaining configurations of Π.

Step 2. As shown in Fig. 3, we apply the acceptance function $\mathcal{A}(\sigma(\pi), \mathrm{Sim}_\pi(\pi))$ provided by $\mathcal{C}$ to discriminate acceptable from unacceptable service configurations, thus obtaining $\Pi' = \{\pi_b, \pi_d\}$. In the figure, we can see that, for instance, π_a is discarded because its similarity with respect to the request descriptor is poor, even

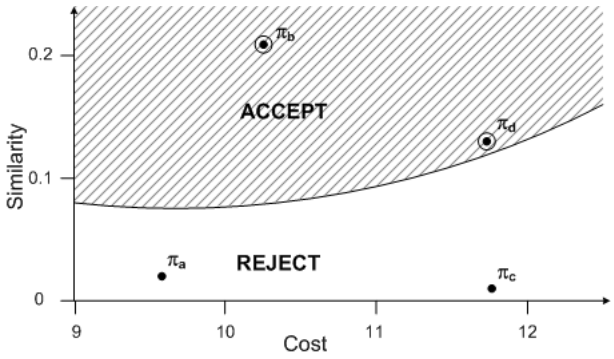

Fig. 3. Eligible configurations and the acceptance function $\mathcal{A}$.

though it has the lowest cost. Differently, π_b falls in the acceptance region, thanks to the good trade-off between cost and similarity.

Step 3.a. Since $\Pi' \neq \varnothing$, $D_C \approx_{\mathcal{A}} D_{\mathcal{P}}$ is true. The final step is performed by $\mathcal{P}$ using its internal policies coded in the choose(Π') function to select a particular configuration π^*. For instance, $\mathcal{P}$ could alternatively select the most similar configuration (π_b), or the most costly one (π_d), or the first one in Π' (again π_b). Assuming $\mathcal{P}$ selects $\pi^* = \pi_b$, the FTP service will give to $\mathcal{C}$ the following service guaranties *GoldStorage* $\wedge$ *SlowBandwidth*.

Additionally, once a configuration is selected, the provider $\mathcal{P}$ may want to fix a crisp value for each resource rather than determining only the service level, e.g., it may want to assign a *SlowBandwidth* of exactly 2 Mbps. This additional step can be performed in several ways depending on the internal policies of $\mathcal{P}$. For instance, $\mathcal{P}$ could defuzzify each $f_i \cap \varphi_i^j$ of π^*, or select a prototype point for each service level, or choose the crisp value with the highest membership degree in each $f_i \cap \varphi_i^j$ of π^*. This is a minor issue and, for the sake of simplicity, we will not enter in further details.

5 Contract Descriptors as Finite State Automata and Transducers

The FAP model cannot express complex policies such as

> Client $\mathcal{C}$ requests **either** a large amount of disk space with an high access throughput **or** a small amount of disk space with no particular limitation on the access throughput.

In order to describe the non-deterministic choice (**either** ... **or** ...) of quality configurations and the dependencies among the service levels of different resources, we extend the previous FAP model. In particular, we use *weighted automata* [9] to define client descriptors and *weighted transducers* [9] to define complex provider descriptors. We start by recalling the definition of semirings taken from [8], which will be used as weights in client automata and provider transducers.

Definition 5.1. *A system* $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ *is a semiring if*

1. $(\mathbb{K}, \oplus, \overline{0})$ *is a commutative monoid with identity element* $\overline{0}$*;*
2. $(\mathbb{K}, \otimes, \overline{1})$ *is a monoid with identity element* $\overline{1}$*;*
3. $\otimes$ *distributes over* $\oplus$*;*
4. $\overline{0}$ *is an annihilator for* $\otimes$ *:* $\forall e \in \mathbb{K}$*,* $e \otimes \overline{0} = \overline{0} \otimes e = \overline{0}$*.*

We will write $\mathbb{K}$ as a shorthand for $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ when the operators are clear from the context. In our framework we use the elements of fuzzy-set semirings to represent service levels. In particular, we choose the semiring whose elements are fuzzy sets and whose operators are the fuzzy intersection $\cap$ and union $\cup$. The identity and annihilator elements are chosen accordingly to the properties of the semirings. For instance, if we consider the semiring $(\mathbb{K}, \cup, \cap, \overline{0}, \overline{1})$, then $\overline{0}$ is the empty fuzzy set of Def. 2.2, while $\overline{1}$ is the fuzzy set with membership value 1 $\forall x \in \mathbb{R}$, where $\mathbb{R}$ is the support of the fuzzy sets.

5.1 Request descriptor

This Section introduces the notion of weighted automata and their utilization as client descriptors.

Definition 5.2. *A client automaton is a 6-tuple* $A_C = (\Sigma, Q, I, F, T, \mathbb{K})$ *where*

1. Σ *is the finite input alphabet;*
2. Q *is a finite set of states;*
3. $I \subseteq Q$ *is the set of the initial states;*
4. $F \subseteq Q$ *is the set of the final states;*
5. $T \subseteq Q \times \Sigma \times \mathbb{K} \times Q$ *is a finite set of weighted transitions;*
6. $\mathbb{K}$ *is a semiring over which transitions weights are defined.*

Given a transition $t_i \in T$, $p(t_i)$ denotes the origin of t_i and $n(t_i)$ the destination. For instance, let t_i be defined as follows

$$t_i = (q_i, x_i, f_i^l, q_i') \text{ with } q_i, q_i' \in Q, x_i \in \Sigma, f_i^l \in \mathbb{K} \tag{3}$$

then $p(t_i) = q_i$ and $n(t_i) = q_i'$. A path $\pi = t_1 \dots t_K \in T^*$ is defined as the composition of transitions $t_i \in T$ such that $n(t_{i-1}) = p(t_i)$ with $i = 2, \dots, K$. We extend the definitions of $p(\cdot)$ and $n(\cdot)$ to paths such that $p(\pi) = p(t_1)$ and $n(\pi) = n(t_K)$. In addition, we define a *labeling function* $\lambda : T \to \Sigma \times \mathbb{K}$ over a transition $t_i = (q_i, x_i, f_i^l, q_i')$ such that $\lambda(t_i) = (x_i | f_i^l)$. The labeling function can be extended to paths by defining $\lambda(\pi) = \lambda(t_1) \circ \dots \circ \lambda(t_K)$, where $\circ$ is the concatenation operator.

Consider the automaton in Fig. 4, where $\Sigma = \{x_1, x_2, x_3, x_4, x_5\}$, $Q = \{q_0, \dots, q_5\}$, $I = \{q_0\}$, $F = \{q_5\}$, $f_j^l \in \mathbb{K}$, and $T = \{(q_0, x_1, f_1^1, q_2), (q_0, x_2, f_1^2, q_2), \dots, (q_4, x_5, f_2^5, q_5)\}$. The labeling function applied on $\pi = t_1 t_3 t_5$ produces $\lambda(\pi) = (x_1 | f_1^1)(x_3 | f_3^1)(x_5 | f_5^1)$.

Similarly to [4], we define the acceptance set Acp for the client automaton A_C as the set of weighted strings generated by the labeling function λ on all paths π leading from an initial state $q_0 \in I$ to a final state $q_f \in F$, that is

$$Acp(A_C) = \{\lambda(\pi) |\ \pi \text{ is a path in } A_C,\ p(\pi) \in I,\ n(\pi) \in F\}. \tag{4}$$

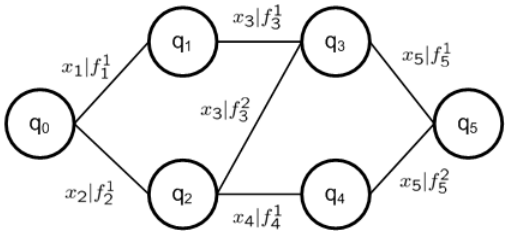

Fig. 4. A client automaton.

In our interpretation, each automaton A_C represents a service request from a client C. The input alphabet of A_C contains the names of service qualities or resources, while $\mathbb{K}$ is the set of fuzzy sets that are used to define the service levels requested by the client. In this interpretation, the set $Acp(A_C)$ defines all the possible compositions of qualities that are acceptable for the client C together with a soft measure of the service level requested for each service.

Now, we refine the Def. 4.2 of the client descriptors as

Definition 5.3 (Client descriptor). *A client descriptor D_C is a couple $(\mathbb{F}, \mathcal{A})$. $\mathbb{F}$ is the acceptance set $Acp(A_C)$ of the acyclic weighted automaton $A_C = \{\Sigma, Q, I, F, T, \mathbb{K}\}$, having transitions $t_i \in T$ of the form (q_i, x_i, f_i^l, q_i'), where each $f_i^l \in \mathbb{K}$ is a fuzzy set defined on the universe of discourse S_i of the linguistic variable named $x_i \in \Sigma$. $\mathcal{A}$ is an acceptance function $U \times [0,1] \rightarrow \{0,1\}$.*

Consequently, clients may define more expressive contract descriptors in which the set $\mathbb{F}$ of alternative desired configurations is defined by a weighted automaton.

Example 5.1. Consider the automaton in Fig. 5 defining a request descriptor for the FTP service. The choice between a *BasicStorage* service and a *LargeStorage* service is expressed by the two transitions that start from the initial state q_0. Moreover, this automaton expresses that the requested level for the bandwidth depends on level offered for the storage. In particular, the client is satisfied by a large-storage service only if it is delivered together with a fast bandwidth access (path $q_0 \rightarrow q_1 \rightarrow q_3$ in Fig. 5), whereas it can accept a slow bandwidth service if it is offered in conjunction with a basic-storage service (path $q_0 \rightarrow q_2 \rightarrow q_4$). Note that the client requires different levels for the same resource depending on the path followed in the automaton. For instance, the two transitions in Fig. 5 ending in q_3 associate two different fuzzy sets ($Fast^1$ and $Fast^2$) to the same resource *Bandwidth*.

5.2 The Provider Descriptor

In what follows we recall the definition of weighted finite state transducer (FST) of [9], which will be used for formalizing the notion of provider descriptors.

Definition 5.4. *A weighted finite state transducer is a 7-tuple*

$$FST_{\mathcal{P}} = (\Sigma_{\mathcal{P}}, C_{\mathcal{P}}, Q_{\mathcal{P}}, I_{\mathcal{P}}, F_{\mathcal{P}}, T_{\mathcal{P}}, \mathbb{K}),$$

where

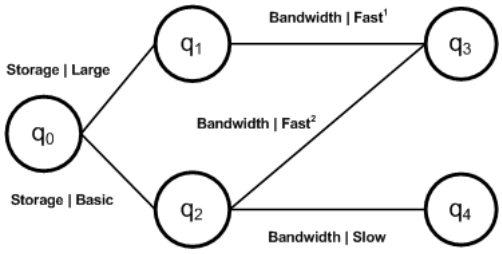

Fig. 5. Client automaton for the FTP service example.

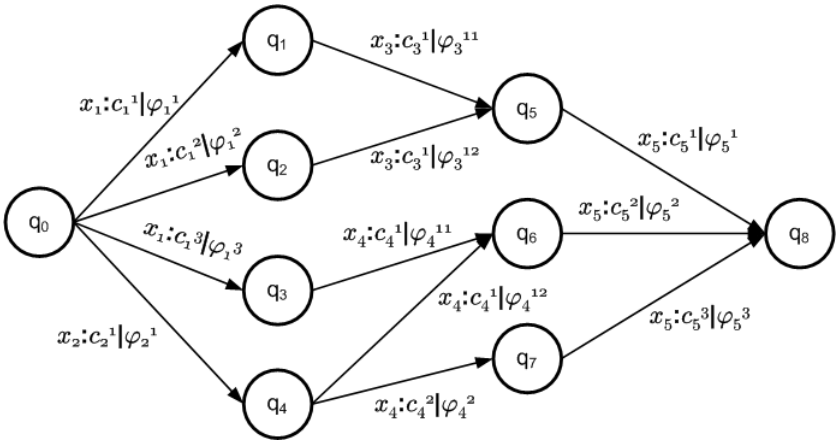

Fig. 6. A server transducer.

1. $\Sigma_{\mathcal{P}}$ *is the finite input alphabet;*
2. $C_{\mathcal{P}}$ *is the finite output alphabet;*
3. $Q_{\mathcal{P}}$ *is a finite set of states;*
4. $I_{\mathcal{P}} \subseteq Q_{\mathcal{P}}$ *is the set of the initial states;*
5. $F_{\mathcal{P}} \subseteq Q_{\mathcal{P}}$ *is the set of the final states;*
6. $T_{\mathcal{P}} \subseteq Q_{\mathcal{P}} \times \Sigma_{\mathcal{P}} \times C_{\mathcal{P}} \times \mathbb{K} \times Q_{\mathcal{P}}$ *is a finite set of weighted transductions from the input alphabet* $\Sigma_{\mathcal{P}}$ *to the output alphabet* $C_{\mathcal{P}}$*;*
7. $\mathbb{K}$ *is a semiring over which transductions weights are defined.*

A weighted transducer is shown in Fig. 6, where the input alphabet is $\Sigma_{\mathcal{P}} = \{x_1, x_2, x_3, x_4, x_5\}$, the states are $Q_{\mathcal{P}} = \{q_0, \ldots, q_8\}$, the initial states are $I_{\mathcal{P}} = \{q_0\}$, the final states are $F_{\mathcal{P}} = \{q_8\}$ and the semiring weights are $\phi_l^{ij} \in \mathbb{K}$, where i denotes the i-th element of the input alphabet, l is the index of the l-th output symbol for a given input symbol and j (optional) is used to differentiate semiring weights for the same combination of input-output pairs.

Given a weighted FST, a path π is the composition of transitions (or transductions) $td_i \in T_{\mathcal{P}}$. We also use the operators $n(\cdot)$ and $p(\cdot)$ to refer to the origin and destination of transductions and transduction paths. The labeling function $\lambda(\cdot)$ is defined such that

$\lambda(td_i) = x_i$ for a transition $td_i = (q_i, c_i^l, x_i, \varphi_i^l, q_i') \in T_s$, and $\lambda(\pi) = \lambda(t_1) \circ \ldots \circ \lambda(t_K)$ for a path $\pi = t_1 \ldots t_K$. In addition, the set of the paths from q to q' is

$$\mathbb{P}(q, q') = \{\pi | p(\pi) = q, n(\pi) = q'\}, \tag{5}$$

while the set of paths from q to q' for the label $x \in \Sigma^*$ is

$$\mathbb{P}(q, x, q') = \{\pi | p(\pi) = q, n(\pi) = q',\ \lambda(\pi) = x\}. \tag{6}$$

Then, the provider descriptor is expressed in terms of a weighted transducer by refining the Definition 4.1.

Definition 5.5 (Provider descriptor). *A provider descriptor $D_{\mathcal{P}}$ is a couple $(FST_{\mathcal{P}}, \mathbb{C})$ where $FST_{\mathcal{P}} = (\Sigma_{\mathcal{P}}, C_{\mathcal{P}}, Q_{\mathcal{P}}, I_{\mathcal{P}}, F_{\mathcal{P}}, T_{\mathcal{P}}, \mathbb{K})$ is a weighted transducer, and where*

1. *$\Sigma_{\mathcal{P}}$ is the set containing the names x_i of the linguistic variables (or resources);*
2. *$C_{\mathcal{P}}$ is the set of linguistic terms c_i^l corresponding to service levels;*
3. *$\mathbb{K}$ is the semiring whose elements are fuzzy sets;*
4. *$\mathbb{C}$ is the set of cost projection functions $\Downarrow_i$.*

As for FAP, we have that $\forall i = 1, \ldots, K \wedge \forall l = 1, \ldots, k_i \wedge \forall j = 1, \ldots, k_l'$: $\varphi_i^{l,j} \in \mathbb{K}$ is a fuzzy set defined on the universe of discourse S_i of the linguistic variable named $x_i \in \Sigma_{\mathcal{P}}$ and related to the linguistic term $c_i^l \in C_{\mathcal{P}}$ representing a class of service.

5.3 The decision procedure

In this new setting, we use the $FST_{\mathcal{P}}$ of a provider descriptor to translate the request made by a client $\mathcal{C}$ to the service levels offered by the provider $\mathcal{P}$. Note that any weighted string x in the request descriptor denotes a possible quality configuration desired by the client. In particular, each $(x_i|f_i) \in x$ requires the resource x_i with quality level f_i^l. Conversely, a transduction $t_i = (q_i, c_i^l, x_i, \varphi_i^j, q_i')$ describes an offer of the resource x_i with the service level c_i^l defined by the fuzzy set φ_i^j. More formally, c_i^l is a linguistic term associated to the linguistic variable x_i, on the universe S_i, such that $\mathcal{M}_i(\varphi_i^l) = c_i^l$. The provider FST translates the client requests, expressed in terms of the input alphabet $\Sigma_s = \Sigma$, to an output alphabet $C_{\mathcal{P}}$ whose elements represent the concrete service levels offered by the provider $\mathcal{P}$. Moreover, the agreement procedure calculates the degree of compliance of requested levels with the offered levels.

The translation from the input alphabet $\Sigma_{\mathcal{P}}$ to the output alphabet $C_{\mathcal{P}}$ is formalised by the following *transduction relation* $\tau : \Sigma \times T_{\mathcal{P}} \longrightarrow C_{\mathcal{P}}$ such that

$$\tau(x_i, td_i) = \begin{cases} c_i^l & \text{if } td_i = (q_i, c_i^l, x_i, \varphi_i^l, q_i') \\ \varepsilon & \text{otherwise} \end{cases} \tag{7}$$

where ε denotes the empty element of the output alphabet. The transduction relation (Equation 7) is extended to strings $x = x_1 \ldots x_K \in \Sigma^*$ and legal paths $\pi \in \mathbb{P}(q_0, x, q_f)$ as follows

$$\tau(x, \pi) = \bigcirc_{i=1}^{K} \tau(x_i, td_i) \tag{8}$$

where $\bigcirc$ denotes the concatenation operator and td_i are the transductions in π. Note that $\tau(x,\pi)$ translates a service request x to the service classes offered by the provider.

Similarly, we define a *weighting function* over strings $x = (x_1|f_i^1)\dots(x_K|f_k^l)$ and legal paths $\pi \in \mathbb{P}(q_0,x,q_f)$ as

$$\sigma(x,\pi) = \delta\left(\bigoplus_{i=1}^{K} \Downarrow_i (f_i^l \otimes \varphi_i^j)\right) \tag{9}$$

where $f_i^l \in \mathbb{K}$ is the service level requested by the client for the resource x_i, $\varphi_i^j \in \mathbb{K}$ is the weight assigning by the transduction $td_i = (q_i, c_i^l, x_i, \varphi_i^j, q_i')$, $\bigoplus$ and $\otimes$ are the operators on the semiring $\mathbb{K}$, $\Downarrow_i$ are the cost functions, while δ is the output function defined as the defuzzyfication operator $\text{cog} : \mathbb{K} \to U$.

Finally, the following *similarity function* calculates the similarity of the requested configuration x with the offered path π as a combination of the similarities between f_i^l and φ_i^j (i.e., the requested and offered levels) for each resource x_i. (Sim is in Equation 1)

$$\text{Sim}_\pi(x,\pi) = \prod_{i=1}^{K} \text{Sim}(f_i^l, \varphi_i^j) \tag{10}$$

The transduction relation in Equation 8 does not take into consideration the transduction weights. We extend its definition to weighted strings $x = (x_1|f_i^1)\dots(x_K|f_k^l)$ and paths $\pi \in \mathbb{P}(q_0,x,q_f)$ by using the weighting and similarity function defined so far, i.e.

$$\tau_w(x,\pi) = \begin{cases} \langle \tau(x,\pi), \sigma(x,\pi), \text{Sim}_\pi(x,\pi)\rangle & \text{if } \text{Sim}_\pi(x,\pi) \neq 0 \\ \langle\cdot\rangle & \text{otherwise} \end{cases} \tag{11}$$

where $\langle\cdot\rangle$ is the empty triplet.

Equation 11 is limited to the transduction of a single string x on a single path π. The full model takes into consideration all the requests modeled by the client automaton A_C, translating them over all eligible paths defined by the provider FST, i.e.

$$R_{FST}(Acp(A_C)) = \cup_{x \in Acp(A_C)} \cup_{\pi \in \mathbb{P}(I_\mathcal{P}, x, F_\mathcal{P})} \tau_w(x,\pi), \tag{12}$$

where $\mathbb{P}(I_\mathcal{P}, x, F_\mathcal{P})$ is the set of paths starting from an initial state $q \in I_\mathcal{P}$, ending in a final state $q' \in F_\mathcal{P}$ and labeled by $x \in \Sigma^*$, that is

$$\mathbb{P}(I_\mathcal{P}, x, F_\mathcal{P}) = \cup_{q \in I_\mathcal{P},\ q' \in F_\mathcal{P}} \mathbb{P}(q,x,q'). \tag{13}$$

Finally, the agreement procedure is formalized as follows.

Definition 5.6 (Evaluation of $\approx_\mathcal{A}$). *Given $D_C = (Acp(A_C), \mathcal{A})$ and $D_\mathcal{P} = (FST, \mathbb{C})$:*

Step 1. *Let $\Pi = R_{FST}(Acp(A_C))$ be the set of all triplets $\langle \tau, \sigma, \text{Sim}\rangle$ representing available service configurations τ having compliance between request and offer expressed by σ and* Sim*.*

Step 2. *Let $\Pi' = \{\langle \tau, \sigma, \text{Sim}\rangle \in \Pi | \mathcal{A}(\sigma, \text{Sim}) = 1\}$ be the set of available service configurations accepted by the client.*

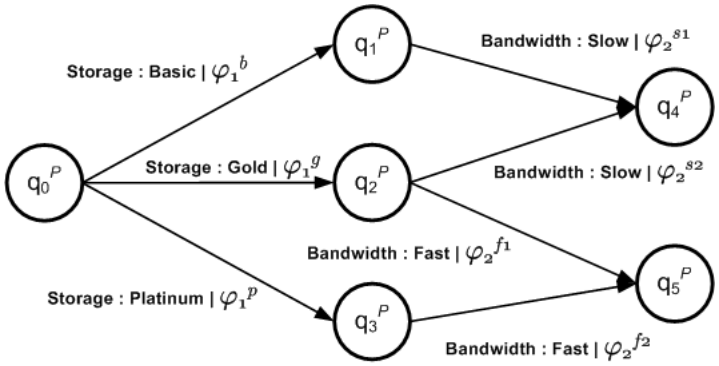

Fig. 7. Provider transducer for the FTP service example.

Step 3.a. *If* $\Pi' \neq \varnothing$ *then* $D_C \approx_{\mathbb{A}} D_{\mathcal{P}}$ *and let* $\pi^* = \text{choose}(\Pi')$ *be the service configuration selected by the provider choice function.*
Step 3.b. *If* $\Pi' = \varnothing$ *then* $(D_C \not\approx_{\mathbb{A}} D_{\mathcal{P}})$.

Example 5.2. Consider the provider descriptor defined in terms of the weighted FST in Fig. 7. The provider can now express complex service allocation policies such as providing a *FastBandwidth* service only to clients requesting either a *GoldStorage* or *PlatinumStorage* service. Moreover, the provider may want to allocate *GoldStorage* clients preferentially to the *FastBandwidth* service: this can be expressed by placing a more restrictive condition on the $q_2^{\mathcal{P}} \to q_4^{\mathcal{P}}$ edge than on the $q_2^{\mathcal{P}} \to q_5^{\mathcal{P}}$ edge, that is reducing the support of $\varphi_2^{s_2}$ with respect to $\varphi_2^{f_1}$.

Consider the client automaton described in Fig. 5, whose acceptance set is

$$Acp(A_C) = \left\{ (St|f_{large})(Bw|f_{fast^1}), (St|f_{basic})(Bw|f_{slow}), (St|f_{basic})(Bw|f_{fast^2}) \right\}$$

where we substitute *St* to *Storage* and *Bw* to *Bandwidth* to ease the notation. The client descriptor is $D_C = (Acp(A_C), \mathcal{A})$, where $\mathcal{A}$ is a suitable acceptance function.

The provider FST is defined on the signature $\Sigma_{\mathcal{P}} = \{St, Bw\}$ and on the output alphabet $C_{\mathcal{P}} = \{Basic, Gold, Platinum\} \cup \{Slow, Fast\}$, where the former set refers to the *Storage* universe and the latter to the *Bandwidth* universe. Moreover, the FST has states $Q_{\mathcal{P}} = \{q_0^{\mathcal{P}}, q_1^{\mathcal{P}}, q_2^{\mathcal{P}}, q_3^{\mathcal{P}}, q_4^{\mathcal{P}}, q_5^{\mathcal{P}}\}$, with $I_{\mathcal{P}} = \{q_0^{\mathcal{P}}\}$ and $F_{\mathcal{P}} = \{q_4^{\mathcal{P}}, q_5^{\mathcal{P}}\}$ as initial and final states, respectively. Hence, sparing the details of the transductions set $T_{\mathcal{P}}$, we obtain the provider-side contract descriptor $D_{\mathcal{P}} = (FST, \{\Downarrow_{St}, \Downarrow_{Bw}\})$, where $\Downarrow_{St}, \Downarrow_{Bw}$ are two suitable cost-projection functions on the *Storage* and *Bandwidth* universes.

By performing **Step 1** of definition 5.6, we generate the set Π of triplets containing, e.g.,

$$\tau_w(x, \pi) = \langle Basic \circ Slow, \text{cog}\left(\Downarrow_{St} (\varphi_1^b \otimes f_{large}) \oplus \Downarrow_{Bw} (\varphi_2^{s1} \otimes f_{fast}^1) \right), \\ \text{Sim}(\varphi_1^b, f_{large}) \cdot \text{Sim}(\varphi_2^{s1}, f_{fast}^1) \rangle$$

which is obtained by considering $x = (St|f_{large})(Bw|f^1_{fast})$ and $\pi = q^{\mathcal{P}}_0 q^{\mathcal{P}}_1 q^{\mathcal{P}}_4$. The operators on the fuzzy set ($\oplus$, $\otimes$), the cost-projection $\Downarrow_i$ and the defuzzyfication functions behave as described in Section 4. Therefore, after the application of **Step 1** we obtain a set of triplets whose elements are a service configuration (e.g. $BasicStorage \wedge SlowBandwidth$) and two reals describing the similarity between the client request and the provider offer. These triplets are filtered by the acceptance function $\mathcal{A}$ as detailed in **Step 2** and, eventually, the provider selects a solution amongst the elements of the acceptable set Π' by means of the choice function in **Step 3** of Definition 5.6.

6 Conclusion and Future Works

This paper presents a general framework for handling SLA negotiation in which agreements rely on the fuzzy approach: required and offered quality levels are described by fuzzy sets. The described framework is at an initial phase in which many aspects deserve further investigation. In particular, we envisage the following lines:

- *Adjustment of provided service levels*: Currently, published service levels do not depend on the state of the provider. Hence, it would be interesting to resort to fuzzy theory tools for allowing the dynamic modification of the shape and position of the fuzzy sets. For instance, linguistic hedges could be used for dynamically adapting a provider descriptor when new contracts are signed, removing such adjustments when contracts are revoked by applying the corresponding inverse linguistic hedge.
- *Contract enforcement*: The SLA-calculus accounts only for SLA negotiation. We plan to extend the framework for dealing with contract enforcement, i.e., to formally explain how clients interact with providers under already signed contracts and how agreed service levels are enforced.
- *Cost models*: We plan to derive a set of off-the-shelf cost models that can help the providers in defining cost functions for their resources. For instance, cost models could be derived from usage or availability of resources. Since cost functions influence the way fuzzy sets are projected in the reference universe, each cost model should be associated with a proper aggregation and defuzzification operator.
- *Automata and transducers properties*: Our definition of weighted automata and transducers differ from the ones that can be found in the literature [9]. We plan to determine suitable composition operators, similar to those described in [9] for standard automata and FST, that can be used to define a modular approach for constructing complex client and provider descriptors by composing simpler service requests and offers.
- *Implementation*: We plan to investigate how the proposed framework can be embedded into current web service infrastructure. In particular, whether the different elements can be mapped to the de facto standards WSDL, SOAP and UDDI.

References

1. R. Babuska. Fuzzy systems, modeling and identification. Technical report, Delft University of Technology, 2001.

2. M. Buscemi and U. Montanari. CC-Pi: A constraint-based language for specifying service level agreements. Manuscript, 2006.
3. D. Dubois and H. Prade. *Fuzzy Sets and Systems - Theory and Applications*. Academic Press, New York, 1980.
4. D. Gorla, M. Hennessy, and V. Sassone. Security policies as membranes in systems for global computing. *Logical Methods in Computer Science*, 1(3), 2005.
5. C.-L. Huang, K.-M. Chao, and C.-C. Lo. A moderated fuzzy matchmaking for web services. In *CIT 2005: Proceedings fo the Fifth International Conference on Computer and Information Technology*, pages 1116 – 1122. IEEE Computer Society, Sep 2005.
6. A. Keller and H. Ludwig. The WSLA framework: Specifying and monitoring service level agreements for web services. *Journal of Network and Systems Management*, 11(1), 2003.
7. M. Lin, J. Xie, H. Guo, and H. Wang. Solving qos-driven web service dynamic composition as fuzzy constraint satisfaction. In *EEE '05: Proceedings of the 2005 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE'05) on e-Technology, e-Commerce and e-Service*, pages 9–14, Washington, DC, USA, 2005. IEEE Computer Society.
8. M. Mohri. Semiring frameworks and algorithms for shortest-distance problems. *Journal of Automata Languages and Combinatorics*, 7(3):321–350, 2002.
9. M. Mohri. Weighted finite-state transducer algorithms: An overview. In C. Martn-Vide, V. Mitrana, and G. Paun, editors, *Formal Languages and Applications*, volume 148 of *Lecture Notes in Computer Science*. Springer, Berlin, 2004.
10. R. De Nicola, G. Ferrari, U. Montanari, R. Pugliese, and E. Tuosto. A formal basis for reasoning on programmable qos. In N. Dershowitz, editor, *Verification: Theory and Practice, Essays Dedicated to Zohar Manna on the Occasion of His 64th Birthday*, volume 2772 of *Lecture Notes in Computer Science*, pages 436–479. Springer, 2003.
11. M. Di Penta and L. Troiano. Using fuzzy logic to relax constraints in GA-based service composition. In *GECCO '05: Proceedings of the 2005 Genetic and Evolutionary Computation Conference*, 2005.
12. M. Setnes, R. Babuska, U. Kaymak, and H. R. van Nauta Lemke. Similarity measures in fuzzy rule base simplification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 28(3):376–386, 1998.
13. Web services agreement specification (ws-agreement). version 2005/09, 2005.
14. L.A. Zadeh. Fuzzy Sets. *Information and Control*, 3(8):338–353, 1965.
15. H.-J. Zimmermann. *Fuzzy set theory and its applications (3rd ed.)*. Kluwer Academic Publishers, Norwell, MA, USA, 1996.