Constraint-Based Policy Negotiation and Enforcement for Telco Services*

Maria Grazia Buscemi
IMT Lucca, Italy

Laura Ferrari Corrado Moiso
Telecom Italia Lab, Italy

Ugo Montanari
University of Pisa, Italy

Abstract

Telco services are evolving under several aspects: for instance, services may combine different telecommunication features (messaging, multi-media, etc.) and may be activated and controlled by applications deployed in a3rd party domain. Telco infrastructures are following this trend by adopting Service Oriented Architecture solutions, e.g. for composing services and for introducing uniform interaction models among services. In a SOA-based system, capabilities, requirements and general features of services can be expressed in terms of policies*. Such policies are negotiated in order to define a Service Level Agreement among the involved parties. In this paper we show how to specify, negotiate, and enforce policies for Telco services by using a constraint-based model, the cc-pi calculus. This language extends concurrent constraint programming with synchronous communication and local names, and with the notion of soft constraints, that generalise classical constraints to represent preference levels. In cc-pi calculus, policies are expressed as soft constraints and the parties involved in the negotiation as communicating processes. The model allows to specify complex scenarios in which policy negotiations and validations can be arbitrarily nested.*

1. Introduction

A recent trend in Telecommunication is to adopt Web services technologies to expose *capabilities* (e.g. call control, sending/receiving messages, access information on end users) implemented in a Telco network to application deployed in 3rd party administrative domains. In such a context, network operators and 3rd parties have to define a Service Level Agreement on Web services in order to monitor the access and usage of Telco capabilities.

*Research supported by the EU IST-FP6 16004 Integrated Project SENSORIA

In a Web Service scenario, *polices* are used to express preferences, requirements, conditions for the control, configuration and protection of capabilities and end-users involved in a service execution. Upon service subscription, the provider and the client negotiate their policies. If the policy negotiation succeeds, the two parties can conclude a contract specifying a certain SLA. In the simplest case, one of the two parties exposes a contract template that the other party can fill in with values in a given range. However, in general the two parties may need a real negotiation in which they pose arbitrary complex policies. Moreover, if the parties fail to reach an agreement, they may substitute their respective policies with less restrictive conditions. During the service execution, the service usage is checked for compliance with the SLA defined at subscription time. In this work we show how to apply a constraint-based model, the cc-pi calculus, for specifying, negotiating, enforcing policies for Telco services.

The cc-pi calculus [5] is a simple model of contracts for QoS and SLAs that also allows to study mechanisms for resource allocation. This language is inspired by two basic programming paradigms: name-passing calculi (see e.g. [7]) and concurrent constraint programming (`cc` programming) [10]. Specifically, the cc-pi calculus combines synchronous communication and a restriction operation à la process calculi with operations for creating, removing and make logical checks on constraints.

In cc-pi, the parties involved in a policy negotiation is modelled as communicating processes and the SLA guarantees and requirements are expressed as constraints that can be generated either by a single process or as a result of the synchronisation of two processes. Moreover, the restriction operator of the cc-pi calculus can limit the scope of names thus allowing for local stores of constraints, which may become global after a synchronisation.

Our constraint-based model relies on the notion of c-semiring. A c-semiring consists of a set equipped with two binary operations, the product $\times$ for com-

bining semiring values and the sum + such that $a + b$ yields the worst value that is better than a and b. C-semirings are very stable, since cartesian products, functional spaces and powerdomains of c-semirings are c-semirings. For this reason, c-semirings allow to model networks of constraints for defining constraint satisfaction problems (CSPs) [8], that is a well-established formalism, especially studied in the artificial intelligence area, adequate to specify many kinds of real-life problems. A single constraint, or even a network of constraints, is a function which, given an assignment of the variables to some domain D, returns a boolean, or rather a value in a generic c-semiring in the case of *soft* constraints. In fact it is easy to define c-semirings expressing fuzzy, hierarchical, or probabilistic values. Also, optimization algorithms work on the c-semiring consisting of the reals plus infinity with the operations of sum as $\times$ and min as $+$. Several efficient algorithms defined for ordinary, crisp constraints, like local propagation or dynamic programming, can be generalized to c-semirings.

In this paper we consider a service called CallBySms to show the applicability of our constraint-based model for specifying policies for Telco Web Services and negotiation mechanisms. The CallBySms service allows a mobile phone user to activate a voice call by sending to a specific service number an SMS with the nickname of the callee. The service is built on two Telco services provided by a network operator, specifying the operations necessary respectively to set-up and control calls and to receive/send SMS messages.

Several approaches for specifying SLA constracts are emerging. SLAng [11] and WSLA [6] are XML-based languages for defining SLAs at a lower level of abstractions. The elements of SLAng are also constraints on the behaviour of associated services and service clients, but their are specified in OCL. WSLA provides the ability to create new SLAs as functions over existing metrics. This is useful to formalise requirements that are expressed in terms of multiple QoS parameters. The semantics for expressions over metrics is not formally defined, though.

2. Telco Architecture

The Telecommunication Services, i.e., the services that are provided by a telecommunication infrastructure managed by a public network operator, are evolving by considering several aspects of *convergence*: (i) convergence of media, i.e. the same service has to combine different types of communications, including voice, video, data streams; (ii) convergence of terminals, i.e. a service should be able to be activated by heterogeneous terminals (including PCs, mobiles, PDAs, TV SetTop-Boxes, etc.) and telecommunication access networks (e.g., ADSL, UMTS, GSM/GPRS), and the same service session could involve different types of terminals; (iii) convergence of service features, i.e. a service may combine different telecommunication features, including multi-media communication, messaging, and content access; (iv) convergence of Telco and Internet/Web worlds, i.e. the distinction among the application contexts is disappearing, e.g. a telecommunication service could be configured, activated, and controlled by Web applications (such as chat applications enabling the activation of phone calls), or an IT application could integrate some Telco features (like fleet management application integrating location features). In general telecommunication services are created, executed and managed by a Service Layer, which consists of a set of systems implementing the functions needed for the service delivery. Most of the current service layers are realized as a set of *vertical* platforms, named Silos, each of them specialized to provide services involving a specific Telco Feature and a specific network. Usually such platforms integrate in a single system the service execution environments with the Telco features and some supporting functions (e.g., payment, authentication, profiles). Since the vertical systems deployed in a service layer are in general loosely integrated, it is quite difficult to share a function across different platforms (i.e., to allow that a service deployed on a platform A can access a function implemented on a platform B) and the same functions are duplicated on several platforms.

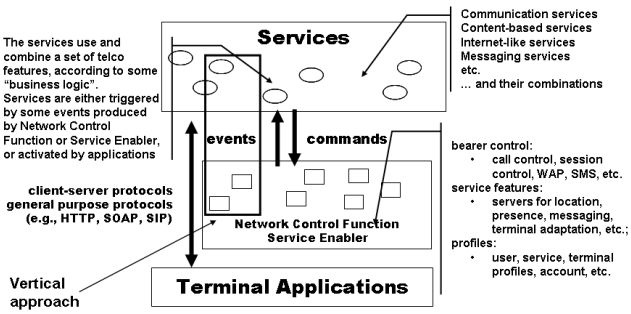

Figure 1. Vertical Telco Architecture

2.1. Telco Architecture versus SOA

The vertical organization of a service layer introduces several problems in dealing with the realization of 'converged' services. In order to improve such situation, the service layer is evolving towards a *horizontal*

approach based on: (i) integration among systems for service delivery which are deployed in the operator infrastructure; (ii) sharing of and interoperability among functions, enablers and service capabilities. A possible trend for the evolution of the service layer according to a horizontal approach is the adoption of Web Services and Service Oriented Architecture solutions. At the moment, there are already several initiatives that consider this possibility, such as the specification of Web Services for controlling Telecommunication services, e.g. Parlay X Web Services [2] jointly specified by ETSI and 3GPP.

A possible model for the horizontal Service Layer is depicted in figure 2. The model includes the following macro-functions:

Service Execution: it contains functions for the deployment and the execution of the business logic of the services; typically the execution functions are carried out by one or more execution environments/application servers based on several technologies (e.g. BPEL [1]).

Service Exposure: it provides functions for secure and controlled interactions among applications deployed in 3rd party domains and the exposed services which are executed in the Service Execution.

Telco Web Services: it implements a uniform set of APIs (e.g., based on an event/command interaction model) that provide an abstract view of the Network Control Functions and Service Enablers. An example of Telco Web Services is provided by the standard specification Parlay X. The Telco Web Services may be used by service logic executed in the Service Execution in order to interact with the Service Enablers and Network Control Function and may be exposed to 3rd parties through the Service Exposure Function.

Service Enablers: they are a set of functions that enable the deployment and the delivery of services based on features additional to the 'basic' connectivity/session control; examples of Service Enablers are: communication enablers for multimedia services (e.g., a conference server), user interaction enablers for service-user/terminals interaction; messaging enablers to control messages exchange; info delivery enablers to control information and content delivery to users via Web, WAP, streaming; and context enablers for context data processing (e.g. a location server).

In particular in this horizontal architecture the Web Services/ SOA approach may be adopted. Figure 3

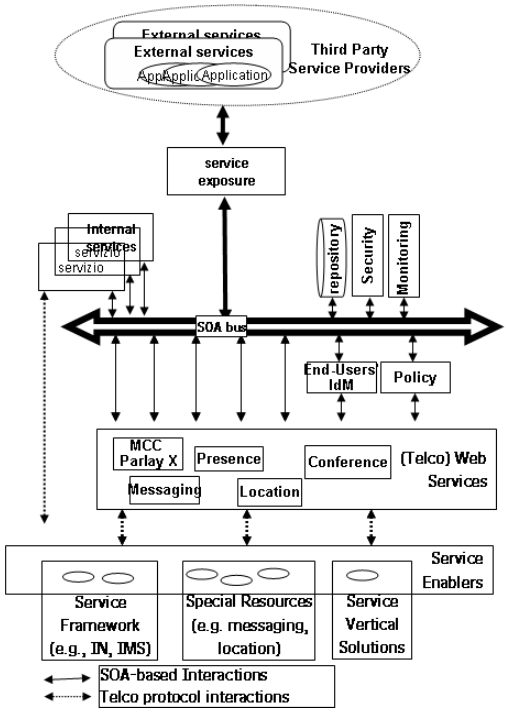

Figure 2. A Possible Model for a Horizontal Service Layer

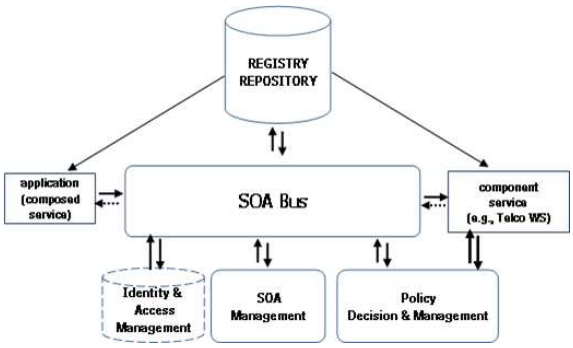

Figure 3. SOA of the Telco Architecture

highlights the main components that are necessary for such an architecture:

Registry/Repository: directory of the services, with all the relevant information useful for their usage and management;

SOA Management: management of the SOA infrastructure (e.g., monitoring of WS usage);

SOA Bus: mediation functions for processing and control on SOA messages, through Intermediaries (e.g., policy enforcement, security checks, rerouting, load-balancing, event notification);

Identity and Access Management: systems to control the access to the services from applications and the involvement of end-users (e.g., privacy);

Policy Decision and Management: handling of policies to control that the usage of enablers by applications fulfills the parameters defined at subscription time, i.e. the SLA. The Policy Decision and Management [3] is meant to support: (i) general purpose policies (enforced by the SOA Bus by message intermediaries); (ii) component service specific policies (enforced by the component implementation); (iii) the possible involvement of external decision points (e.g., end-user account management).

In the rest of the paper we will focus on the last of the above components and we will address one of the critical point of SOA in Telco Service Layer, that is the specification of policies and policy enforcement/negotiation mechanisms.

3. Policies for Telco Web Services

Telecommunication networks are adopting Web Services technologies to expose *capabilities*, such as call control, sending/receiving messages, access information on users, to applications deployed in 3rd party administrative domain. In a SOA-based system, capabilities, requirements and general features of services can be expressed in terms of *policies*. Such policies are negotiated in order to define a Service Level Agreement among the involved parties.

3.1. Telco Policy Description

The reference model of policies for Telco Web Services is reported in Figure 4. The Policy Management System resides in the network operator domain and it is in charge of handling the policies between the 3rd Party Application and the Telco Web Services that compose a certain service, i.e. the policies defined for the specific Parlay X Web Services involved in the service. The policies may apply to the request/response flow or in the notification case. The Policy Management System can validate policies either on interception (the evaluation is performed on the messages of request/response of Web Services) or on invocation (the evaluation is requested from the Web Service implementation).

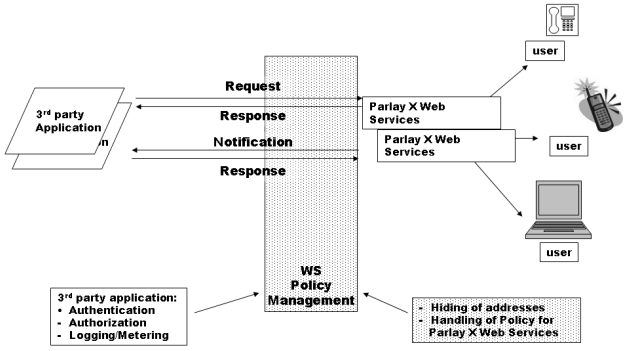

Figure 4. Reference Model for Telco Web Service Policies

Different types of policies can be identified. Below we give a possible categorisation of policies:

policies on authentication concern authentication mechanisms such as IP address, username/password;

policies on authorisation include e.g. access on subscription, charge/free access;

policies on time range are used to verify if a web service or an operation that is part of a web service can be invoked on a certain moment;

policies on invocation frequencies specify, e.g., how many times an operation can be invoked in a certain time range;

policies on end-users' addresses are used to control the validity of the network addresses;

policies on hiding of end-users' addresses indicate if user network addresses can be explicitly passed or if they have to be hidden, e.g., by mapping addresses and aliases;

policies on the session validity is meant to verify whether a request or notification is within a certain session of usage. The session validity check for an application is based on the relationship between a generated session identifier and the application for which the session identifier has been generated (e.g. time validity of the session, number of invocation of an operation in a session).

3.2. CallBySms: An Example of Telco Services

We introduce a service scenario named CallBySms that we will use in subsequent sections to show the applicability of the constraint-based approach. The service is built on the Parlay X Web Services and, in particular, it uses ThirdPartyCall and ShortMessaging services for specifying the operations respectively necessary to set-up and control calls and to receive/send short messages.

The CallBySms service allows a mobile phone user to activate a voice call by sending an SMS message to a specific service number. The SMS message must contain a nickname of the person the user wishes to call. The service is able to automatically find the number associated with the nickname and to set up a third party call between the user and the callee. In order to keep privacy, the service does not know actual phone numbers, but only opaque-id representing users. Figure 5 depicts a possible service scenario in which John wishes to call Mary and he knows that Mary's nickname is "sunshine".

1. The Third Party application subscribes the services of Parlay X WS that compose the CallBySms service and signs a SLA contract with the Parlay X WS;
2. The CallBySMS service is activated and the Third Party application receives a service number, e.g. 11111;
3. Mary sends an SMS "REGISTER sunshine" to the service number 11111;
4. The service associates "sunshine" to the opaque-id of Mary;
5. John sends an SMS "CALL sunshine" to the service number 11111;
6. The service retrieves the opaque-id associated to "sunshine" and set-up a call;
7. John's phone rings; John answers and gets the ringing tone;
8. Mary's phone rings; Mary answers;
9. John and Mary are connected.

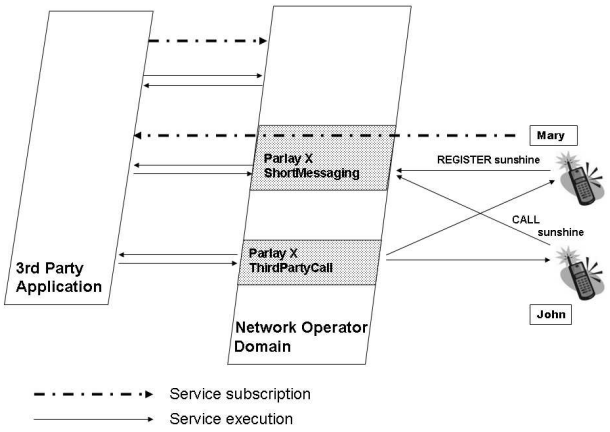

Figure 5. CallBySms Service Scenario

4. Modelling Policies through Constraints

In this section we show how to specify the Telco policies in terms of c-semiring-based constraints. The most interesting policies from the viewpoint of constraint-based formalisations are those ones specifying ranges of admitted values, possibly with different preference levels, or arbitrary formulas, rather than single values. In fact, the latter kind of policies can be trivially combined into a contract and validated, thus not requiring the c-semiring machinery. For this reason, in this work we disregard token-based policies e.g. concerning authorization, authentication or session validity.

Furthermore, in our approach we do not need to distinguish the policy types – as specified by the taxonomy given in § 3 – according to the involved parties. In fact, our model focuses on SLA contracts among arbitrary service providers and clients, e.g. the Parlay X WS offering certain Telco services and a 3rd party application or the application providing a service, possibly combining some Telco services, and an end-user.

We now give the basic definitions and properties concerning c-semirings and named c-semirings. We refer to [4] and [5], respectively, for a more comprehensive treatment.

4.1. Semiring-based Constraints

C-semiring A *constraint semiring (c-semiring)* is a tuple $\langle A, +, \times, 0, 1\rangle$ such that: (i) A is a set and $0, 1 \in A$; (ii) $+$ is commutative, associative, idempotent, 0 is its unit element and 1 is its absorbing element; (iii) $\times$ is associative, commutative and distributes over $+$, 1 is its unit element and 0 is its absorbing element.

Intuitively, the sum $a + b$ chooses the worst constraint better than a and b, while the product $a \times b$ combines two constraints. C-semirings are equipped with a partial ordering $\leq$, which is defined as $a \leq b$ iff $a + b = b$, and which means that a is more constrained than b or, more interestingly, that a entails b, $a \vdash b$.

Typical examples are the c-semiring for classical constraints $\langle\{\mathsf{False}, \mathsf{True}\}, \vee, \wedge, \mathsf{False}, \mathsf{True}\rangle$, the c-semiring for fuzzy CSPs $\langle[0, 1], max, min, 0, 1\rangle$, and the c-semiring for probabilistic CSPs $\langle[0, 1], max, \cdot, 0, 1\rangle$, and the c-semiring of weighted CSPs $\langle[0, \ldots, +\infty], \mathsf{min}, +, 0, +\infty\rangle$. Since the Cartesian product of two c-semirings is still a c-semiring, it is also possible to model multicriteria optimization in this framework.

Named c-semirings Named c-semirings are c-semirings enriched with a notion of name equalities, a permutation algebra structure that identifies the *support* $\mathrm{supp}(c)$ of every element c of the c-semiring, i.e. the (finite) set of names that are *relevant* for c, and a hiding operator $(\nu\, x.\,)$ that makes a name x local in c, in the style of process calculi. Assume a countable set of names $\mathcal{N}$, ranged over by $x, y, z, \ldots$. Formally, a *named c-semiring* $\mathcal{C} = \langle C, +, \times, \nu\, x.\,, \rho, 0, 1\rangle$ is a tuple such that: (i) the name equalities $x{=}y \in C$; (ii) $\langle C, +, \times, 0, 1\rangle$ is a c-semiring; (iii) $\langle C, \rho\rangle$ is a (finite-support) permutation algebra; (iv) $\nu\, x.\, : C \to C$, for each name x, is a unary operation. Furthermore, named c-semirings satisfy a set of axioms ruling how to combine with each other c-semiring operations, such as $+$, $\times$, ν and ρ.

Given a named c-semiring $\langle C, +, \times, \rho, \nu\, x.\,, 0, 1\rangle$, a *(named) constraint* c is an element of C. We write $c(x_1, \ldots, x_n)$ for a constraint c with support $\mathrm{supp}(c) = \{x_1, \ldots, x_n\}$. This notation allows to show the free names of c and to explicitly apply permutations to c. A set of constraints $C' = \{c_1, \ldots, c_n\}$, such that $C' \subseteq C$, is *consistent* if $(c_1 \times \ldots \times c_n) \neq 0$. Moreover, given a constraint $c \in C'$, C *entails* c, written $C \vdash c$, if $(c_1 \times \ldots \times c_n) \leq c$.

Soft CSP Given a domain D of interpretation for $\mathcal{N}$, and a c-semiring $S = \langle A, +, \times, 0, 1\rangle$, a *soft* constraint c can be represented as a function $c = (\mathcal{N} \to D) \to A$ associating to each variable assignment $\eta = \mathcal{N} \to D$ a value of A. Soft constraints can be combined by using the c-semiring operations.

A named c-semiring for soft constraints $\mathcal{C}_{\mathrm{soft}}$ can be defined as the tuple $\mathcal{C}_{\mathrm{soft}} = \langle C, +', \times', \nu\, x.\,, \rho, 0', 1'\rangle$ such that: (i) C is the set of all soft constraints over $\mathcal{N}$, D and S; (ii) name equalities $x{=}y$ are defined as $(x = y)\eta = 1$ if $\eta(x) = \eta(y)$, $(x = y)\eta = 0$ otherwise; (iii) $(c_1 +' c_2)\eta = c_1\eta + c_2\eta$; (iv) $(c_1 \times' c_2)\eta = c_1\eta \times c_2\eta$; (v) $(\nu\, x.\, c)\eta = \sum_{d \in D} (c\eta[{}^{d}\!/\!_{x}])$, where $\sum_{d \in D}$ denotes the c-semiring sum operator and the assignment $\eta[{}^{d}\!/\!_{x}]$ is defined, as usual, as $\eta[{}^{d}\!/\!_{x}](y) = d$ if $x = y$, $\eta(y)$ otherwise; (vi) $(\rho\, c)\eta = c\overline{\eta}$ with $\overline{\eta}(x) = \eta(\rho(x))$; (vii) $0'\,\eta = 0$ and $1'\eta = 1$ for all η.

Of course, $\mathcal{C}_{\mathrm{soft}}$ can be instantiated by considering a specific c-semiring S. For instance, the choice $S = \langle\{\mathsf{False}, \mathsf{True}\}, \vee, \wedge, \mathsf{False}, \mathsf{True}\rangle$ leads to solutions consisting of the set of tuples of legal domain values. The notation $x \leq \mathsf{a} \times \mathsf{b} \leq y$, where x, y are names in $\mathcal{N}$ and a, b are domain values in D, is a compact way for representing the constraint c of C defined as $c = (\mathcal{N} \to D) \to \{\mathsf{False}, \mathsf{True}\}$, with η being an assignment $\eta = \mathcal{N} \to D$, such that $c\eta = \mathsf{True}$ if $\eta(x) \leq \mathsf{a}$ and $\mathsf{b} \leq \eta(y)$, while $c\eta = \mathsf{False}$ otherwise.

4.2. Policies for CallBySms

We now show how to formalise and combine policies for Telco Web Services in terms of semiring-based constraints by focusing on the CallBySms service introduced in the previous section. Particularly, we define two kinds of policies concerning *time* and *frequency*. For simplicity, in the following we take the reference constraint system to be a CSP by considering the named c-semiring $\mathcal{C}_{\mathrm{soft}}$ over the c-semiring $S = \langle\{\mathsf{False}, \mathsf{True}\}, \vee, \wedge, \mathsf{False}, \mathsf{True}\rangle$. However, such constraint system can be easily generalised to soft constraints by replacing S with an arbitrary c-semiring.

Time policies Suppose the Parlay X Web Services (ParX) guarantees to set-up third party calls at any time in the initial time range $[7, \ldots, 9]$ and final time range $[15, \ldots, 18]$. This policy can be expressed by the constraint $c_{\mathrm{time}}(i, f) = (7 \leq i \leq 9) \times (15 \leq f \leq 18)$. Similarly, the 3rd party application (3rdPA) requires that phone calls are set-up at any time in the initial and final time ranges $[6, \ldots, 8]$ and $[17, \ldots, 19]$. This policy can be represented by the constraint $d_{\mathrm{time}}(i, f) = (6 \leq i \leq 8) \times (17 \leq f \leq 19)$. The result of combining these policies is the intersection of the initial and final time ranges. In terms of constraints this is given by the c-semiring product $e_{\mathrm{time}}(i, f) = c_{\mathrm{time}}(i, f) \times d_{\mathrm{time}}(i, f) = (7 \leq i \leq 8) \times (17 \leq f \leq 18)$.

Frequency policies Suppose ParX wants to pose a bound max_call on the number nc of phone calls that can be set-up upon request of end users while the CallBySms service is active. This condition can be expressed by the constraint $c_{\text{freq}}(nc) = 0 \leq nc \leq$ max_call. On the other side, in order to ensure that each registered user receives a maximum call_per_pers of calls, 3rdPA requires a maximum max_call/call_per_pers of registration requests nr. The above policy can be represented by the constraint $d_{\text{freq}}(nr) = 0 \leq nr \leq$ max_call/call_per_pers.

The policies e_{time} and c_{freq} are part of the SLA contract among ParX and 3rdPA and they are validated by the Policy Manager residing within the network operator domain, while the policy d_{freq} depends on the ParX parameter `max_call` but concerns the agreement of 3rdPA with every end-user and it is checked by 3rdPA.

5. Negotiation and Enforcement in Cc-pi

In this section we employ a language based on c-semirings, the cc-pi calculus [5], as a framework for specifying, negotiating and validating Telco policies. In this model, the parties involved in a policy negotiation are modelled as communicating processes. First, we outline the basic features of the calculus. We refer to [5] for details.

Syntax Assume the set of names $\mathcal{N}$, ranged over by $x, y, z, \ldots$ and a set of process identifiers, ranged over by D. We let c range over constraints of an arbitrary named c-semiring $\mathcal{C}$. The syntax of the calculus is specified in Table 1. The τ prefix stands for a silent action, the output prefix $\overline{x}\langle\widetilde{y}\rangle$ for emitting over the port x the message $\widetilde{y}$ and the input prefix $x\langle\widetilde{y}\rangle$ for receiving over x a message and binding it to $\widetilde{y}$. Prefix `tell` c generates a constraint c and puts it in parallel with the other constraints, if the resulting parallel composition of constraints is consistent; `tell` c is not enabled otherwise. Prefix `ask` c is enabled if c is entailed by the set of constraints in parallel. Prefix `retract` c removes a constraint c, if c is present. Prefix `check` c is enabled if c is consistent with the set of constraints in parallel. *Unconstrained processes* U are essentially processes that can only contain constraints c in prefixes `tell` c, `ask` c, `retract` c, and `check` c. As usual, 0 stands for the inert process and $U \,|\, U$ for the parallel composition. $\sum_i \pi_i.U_i$ denotes an external choice in which some guarded unconstrained process U_i is chosen when the corresponding guard π_i is enabled. Restriction $(x)\,U$ makes the name x local in U. A defining equation for a process identifier D is of the form $D(\tilde{x}) \stackrel{\text{def}}{=} U$ with $|\tilde{x}| = |\tilde{y}|$. *Constrained processes* P are defined like unconstrained processes U but for the fact that P may have constraints c in parallel with processes. We simply write processes to refer to constrained processes.

Structural Congruence The *structural congruence* relation $\equiv$ is defined as the least congruence over processes closed with respect to α-conversion and satisfying the following rules. Note that the notion of *free names* $\text{fn}(P)$ of a process $P$is extended to handle constraints by stating that the set of free names of a constraint c is the support $\text{supp}(c)$ of c.

$$P|0 \equiv P \qquad P|Q \equiv Q|P \qquad (P|Q)|R \equiv P|(Q|R)$$
$$U + V \equiv V + U \qquad (U + V) + W \equiv U + (V + W)$$
$$(x)0 \equiv 0 \qquad P|(x)Q \equiv (x)(P|Q) \;\; \textit{if} \;\; x \notin \text{fn}(P)$$
$$(x)(y)P \equiv (y)(x)P \qquad D(\tilde{y}) \equiv [\tilde{y}/\tilde{x}]U \;\; \textit{if} \;\; D(\tilde{x}) \stackrel{\text{def}}{=} U$$

These axioms can be applied for reducing every process P into a normal form $(x_1) \ldots (x_n)\,(C \,|\, U)$, where C is a parallel composition of constraints and U is an unconstrained process.

Reduction Semantics The *reduction relation* over processes $\rightarrow$ is the least relation satisfying the inference rules in Table 2. In the table we adopt the following notations: C stands for the parallel composition of constraints $c_1 \,|\, \ldots \,|\, c_n$; C *consistent* means $(c_1 \times \ldots \times c_n) \neq 0$; $C \vdash c$ if $(c_1 \times \ldots \times c_n) \leq c$; $C - c$ stands for $c_1 \,|\, \ldots \,|\, c_{i-1} \,|\, c_{i+1} \,|\, \ldots \,|\, c_n$ if $c = c_i$ for some i, while $C - c = C$ otherwise.

The idea behind this reduction relation is to proceed as follows. First, rearranging processes into the normal form $(x_1) \ldots (x_n)\,(C \,|\, U)$ by means of rule (STRUCT). Next, applying the rules (TELL), (ASK), (RETRACT), and (CHECK) for primitives on constraints and the rule (COM) for synchronising processes. Finally, closing with respect to parallel composition and restriction ((PAR), (RES)). More in detail, rule (TELL) states that if $C \,|\, c$ is consistent then a process can place c in parallel with C, the process is stuck otherwise. Rules (ASK) and (CHECK) specify that a process starting with an `ask` c or, respectively, `check` c prefix evolves to its continuation if c is entailed by C or, respectively, if $c \,|\, C$ is consistent, and that the process is stuck otherwise. By rule (RETRACT) a process can remove c if c is among the syntactic constraints in C; e.g., the process $x = y \,|\, y = z \,|\,$ `retract` $x = z.U$ does not affect $x = y \,|\, y = z$. In rules (COM), we write

PREFIXES	π	::=	$\tau \mid \overline{x}\langle\tilde{y}\rangle \mid x\langle\tilde{y}\rangle \mid \texttt{tell}\ c \mid \texttt{ask}\ c \mid \texttt{retract}\ c \mid \texttt{check}\ c$
UNCONSTRAINED PROCESSES	U	::=	$\mathbf{0} \mid U\|U \mid \sum_i \pi_i.U_i \mid (x)U \mid D(\tilde{y})$
CONSTRAINED PROCESSES	P	::=	$U \mid c \mid P\|P \mid (x)P$

Table 1. Syntax

(TAU) $C \,|\, \tau.U \to C \,|\, U$ (TELL) $C \,|\, \texttt{tell}\ c.U \to C \,|\, c \,|\, U$ if $C \,|\, c$ consistent

(ASK) $C \,|\, \texttt{ask}\ c.U \to C \,|\, U$ if $C \vdash c$ (RETRACT) $C \,|\, \texttt{retract}\ c.U \to (C - c) \,|\, U$

(CHECK) $C \,|\, \texttt{check}\ c.U \to C \,|\, U$ if $C \,|\, c$ consistent

(COM) $C \,|\, (\overline{x}\langle\widetilde{y}\rangle.U + \sum_i \pi_i.U_i) \,|\, (z\langle\widetilde{w}\rangle.V + \sum_j \pi'_j.V_j) \longrightarrow C \cup \{\widetilde{y} = \widetilde{w}\} \,|\, U \,|\, V$
if $|\widetilde{y}| = |\widetilde{w}|$, $C \,|\, \widetilde{y} = \widetilde{w}$ consistent and $C \vdash x = z$

(SUM) $$\frac{C \,|\, \pi_i.U_i \to P \text{ for some } i}{C \,|\, \sum_i \pi_i.U_i \to P}$$ (PAR) $$\frac{P \to P'}{P \,|\, U \to P' \,|\, U}$$

(RES) $$\frac{P \to P'}{(x)\,P \to (x)\,P'}$$ (STRUCT) $$\frac{P \equiv P' \quad P' \to Q' \quad Q' \equiv Q}{P \to Q}$$

Table 2. Reduction semantics

$\widetilde{y} = \widetilde{w}$ to denote the parallel composition of fusions $y_1 = w_1 \,|\, \ldots, \,|\, y_n = w_n$. Intuitively, two processes $\overline{x}\langle\widetilde{y}\rangle.P$ and $z\langle\widetilde{w}\rangle.Q$ can synchronise if the equality of the names x and z is entailed by C and if the parallel composition $C \,|\, \widetilde{y} = \widetilde{w}$ is consistent. Note that it is legal to treat name equalities as constraints c over $\mathcal{C}$, because named c-semirings contain fusions. Rule (PAR) allows for closure with respect to unconstrained processes in parallel. This rule imposes to take into account all constraints in parallel when applying the rules for constraints and synchronisation.

The present semantics does not specify how to solve at each step the constraint system given by the parallel composition of constraints C. However, in [9] it is shown how to apply dynamic programming to solve a CSP by solving its subproblems and then by combining solutions to obtain the solution of the whole problem. A visual representation of the problem is given by considering a graph where names are represented as nodes and constraints as arcs connecting the names involved in each constraint.

5.1. Modelling CallBySms

In Table 3 we show the formal specification in cc-pi of the policy negotiation and service execution scenario of CallBySms. On top of the table we recall the time and frequency policies expressed in terms of constraints. The negotation phase between 3rdPA and ParX consists of the two parties placing their own constraints and trying to synchronise on port x in order to export their local parameter plus two variables for marking the beginning and conclusion of the service execution. If the set of all such constraints, including the name equalities induced by the synchronisation, is consistent, the two parties have concluded a contract, which is expressed by the c-semiring product $\times$ of all constraints. The process $Clock_{\mathsf{T}}$ is meant to simulate the actual time by increasing of a certain amount a time variable t starting from the initial value T. We assume this + operation automatically resets the clock by the end of the day. Upon an execution request, *ParX* is ready to accept requests and to forward them to 3rdPA.

An end-user intending to register to CallBySms tries to synchronise with *ParX* on port z with its private identity *mary* and nickname *sunshine* as parameters, and waits for an acknowledgment on *mary*. *ParX* forwards this request to *3rdPA* by sending on x the nickname and a private channel name *ch*, though not revealing the user identity. *3rdPA* checks whether the number of registered users is within the agreed bound and increases the respective counter (process $Verify_{\mathsf{R}}$), before accepting the request. Then, *ParX* notifies the

POLICIES

$$
\begin{aligned}
c_{\text{time}}(i,f) &= (7 \le i \le 9) \times (15 \le f \le 18)\\
c_{\text{freq}}(nc) &= 0 \le nc \le \mathsf{max\_call}\\
d_{\text{time}}(i',f') &= (6 \le i' \le 8) \times (17 \le f' \le 19)\\
d_{\text{freq}}(nr,nc') &= 0 \le nr \le nc'/\mathsf{call\_per\_pers}
\end{aligned}
$$

3RDPA-PARX NEGOTIATION

$$
\begin{aligned}
\text{ParX\_Neg}(x,z) &= (i,f,nc,beg)\,(\texttt{tell}\ c_{\text{time}}(i,f) \times c_{\text{freq}}(nc).\, x\langle i,f,nc,beg\rangle.0)\\
\text{3rdPA\_Neg}(x) &= (i',f',nc',beg',nr)\,(\texttt{tell}\ d_{\text{time}}(i',f') \times d_{\text{freq}}(nr,nc').\\
&\quad \overline{x}\langle i',f',nc',beg'\rangle.0)
\end{aligned}
$$

CLOCK

$$
\text{Clock}_{\mathsf{T}}(t) = \texttt{retract}\ (t=\mathsf{T}).\,\texttt{tell}\ (t=\mathsf{T}+1).\,\text{Clock}_{\mathsf{T}+1}(t)
$$

SERVICE EXECUTION

$$
\begin{aligned}
\text{ParX\_Ex}_{\mathsf{N}}(x,z,i,f,t,nc,beg) &= beg\langle\rangle.\text{ParX\_Acpt\_Reqst}_{\mathsf{N}}(x,z,i,f,t,nc)\\
\text{3rdPA\_Ex}_{\mathsf{R}}(x,beg',nr) &= \overline{beg'}\langle\rangle.\text{3rdPA\_Acpt\_Reqst}_{\mathsf{R}}(x,nr)
\end{aligned}
$$

HANDLING REGISTRATION REQUESTS

$$
\begin{aligned}
\text{Regist\_User}(z,sunshine) &= (mary)\,(\overline{z}\langle mary,sunshine\rangle.\,mary\langle\rangle.\,\text{Wait\_Calls})\\
\text{ParX\_Acpt\_Reqst}_{\mathsf{N}}(x,z,i,f,t,nc) &= (id,nn,ch)(z\langle id,nn\rangle.\overline{x}\langle nn,ch\rangle.\overline{id}\langle\rangle.(\\
&\quad \text{ParX\_Acpt\_Reqst}_{\mathsf{N}}(x,z,i,f,t,nc)\,|\\
&\quad \text{ParX\_Acpt\_Call}_{\mathsf{N}}(i,f,t,nc,id,nn,ch)))\\
\text{3rdPA\_Acpt\_Reqst}_{\mathsf{R}}(x,nr) &= (nn',ch')\,(\text{Test}_{\mathsf{R}}(nr).\,x\langle nn',ch'\rangle.\\
&\quad (\text{3rdPA\_Acpt\_Reqst}_{\mathsf{R}-1}(x,nr)\,|\,\text{3rdPA\_Acpt\_Call}(ch')))
\end{aligned}
$$

HANDLING CALL REQUESTS

$$
\begin{aligned}
\text{Wait\_Calls}(mary) &= (cal')\,(mary\langle cal'\rangle.cal'\langle\rangle.\text{Wait\_Calls})\\
\text{Caller}(sunshine) &= (john)\overline{sunshine}\langle john\rangle.\overline{john}\langle\rangle.0\\
\text{ParX\_Acpt\_Call}_{\mathsf{N}}(i,f,t,nc,id,nn,ch) &= (cal)\,(\texttt{check}\ (i \le t \le f).\text{Test}_{\mathsf{N}}(nc).nn\langle cal\rangle.\\
&\quad \overline{ch}\langle\rangle.\overline{id}\langle cal\rangle.\text{ParX\_Acpt\_Call}_{\mathsf{N}-1}(i,f,t,nc,id,nn,ch))\\
\text{3rdPA\_Acpt\_Call}(ch') &= ch'\langle\rangle.\text{3rdPA\_Acpt\_Call}(ch')
\end{aligned}
$$

TEST AND DECREASE

$$
\text{Test}_{\mathsf{F}}(y) = \texttt{check}\ (0 \le y \le \mathsf{F}).\,\texttt{retract}\ (0 \le y \le \mathsf{F}).\,\texttt{tell}\ (0 \le y \le (\mathsf{F}-1))
$$

SYSTEM

$$
\begin{aligned}
S &= (t,x,z,sunshine)(\text{ParX\_Neg}(x)\,|\,\text{3rdPA\_Neg}(x)\,|\\
&\quad \text{ParX\_Ex}_{\mathsf{max\_call}}(x,z,i,f,t,nc,beg,end)\,|\\
&\quad \text{3rdPA\_Ex}_{\mathsf{max\_call}/\mathsf{call\_per\_pers}}(x,beg',end',ncp,nr)\,|\\
&\quad \text{Regist\_User}(z,sunshine)\,|\,\text{Caller}(sunshine)\,|\\
&\quad \texttt{tell}\ (t=0).\,\text{Clock}_0(t))
\end{aligned}
$$

Table 3. CallBySms specification in cc-pi

successful registration to the user.

A user who wants to call Mary but only knows her nickname is specified by a process sending its private name *john* on the public port *sunshine* and then waiting to be connected with *sunshine* on port *john*. *ParX* verifies that the request is within the legal time range and that the number of calls that can be set-up is under the bound max_call. In case of success, *ParX* send an acknowlegment to *3rdPA* and then forwards the name *john* to the private port *mary* in order to connect the two users.

The whole system S is given by the parallel composition of the processes specifying the policy negotiation, the service execution, the initialised clock and the two users. Note that our framework can be employed to model more complex negotiation scenarios, e.g. in which there is an arbitrary number of end-users or in which *3rdPA* and *ParX* may want to `retract` their initial policies and replace them with less restrictive constraints, in order to reach an agreement.

6. Concluding Remarks

In this paper we have applied the cc-pi calculus, a constraint-based model for specifying SLA contracts, to the Telecommnunication context. Specifically, we have shown how to negotiate and enforce policies for Telco Web services in cc-pi. This application could benefit from further work on the calculus, such as the development of an implementation and of a richer theory of the calculus, e.g. the study of suitable mechanisms for assuring transactional and security properties of process executions. It would also be interesting to compare our formalisation with specifications of Telco policies in XML-based languages for defining SLAs, like SLAng [11] and WSLA [6].

References

[1] Web Services Business Execution Language – Version 2.0.

[2] Open Service Access (OSA) 3GPP. Parlay X Web Services (Release 6). TS 29.199 v6.x.y.

[3] Open Mobile Alliance. Policy evaluation, enforcement and management requirements - Draft Version 1.0.

[4] S. Bistarelli, U. Montanari, and F. Rossi. Semiring-based constraint satisfaction and optimization. *Journal of the ACM*, 44(2):201–236, 1997.

[5] M. G. Buscemi and U. Montanari. Cc-pi: A constraint-based language for specifying service level agreements. In *Proc. ESOP '07*. To appear.

[6] A. Keller and H. Ludwig. The WSLA framework: Specifying and monitoring service level agreements for web services. *Jour. Net. and Sys. Manag.*, 11(1):57–81, 2003.

[7] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, I and II. *Inform. and Comput*, 100(1):1–40,41–77, 1992.

[8] U. Montanari. Networks of constraints: fundamental properties and application to picture processing. *Information Science*, 7:95–132, 1974.

[9] U. Montanari and F. Rossi. Constraint relaxation may be perfect. *Artif. Intell.*, 48(2):143–170, 1991.

[10] V. Saraswat and M. Rinard. Concurrent constraint programming. In *Proc. POPL'90*. ACM Press, 1990.

[11] J. Skene, D. Lamanna, and W. Emmerich. Precise service level agreements. In *Proc. ICSE'04*, 2004.