# Table of Contents

## Research Papers – Long

### Business Process Modeling

Computing Degree of Parallelism for BPMN Processes ............... 1
*Yutian Sun and Jianwen Su*

State Propagation in Abstracted Business Processes .................. 16
*Sergey Smirnov, Armin Zamani Farahani, and Mathias Weske*

Push-Enabling RESTful Business Processes ......................... 32
*Cesare Pautasso and Erik Wilde*

### Quality of Service 1

QoS Analysis for Web Service Compositions Based on Probabilistic QoS ......................................................... 47
*Huiyuan Zheng, Jian Yang, Weiliang Zhao, and Athman Bouguettaya*

Constraint-Based Runtime Prediction of SLA Violations in Service OrCHEstrations .................................................... 62
*Dragan Ivanović, Manuel Carro, and Manuel Hermenegildo*

Optimizing Decisions in Web Services OrCHEstrations ............ 77
*Ajay Kattepur, Albert Benveniste, and Claude Jard*

### Formal Methods

Decidability Results for Choreography Realization ................. 92
*Niels Lohmann and Karsten Wolf*

Conformance Testing for Asynchronously Communicating Services .... 108
*Kathrin Kaschner*

Programming Services with Correlation Sets .......................... 125
*Fabrizio Montesi and Marco Carbone*

Verification of Deployed Artifact Systems via Data Abstraction .... 142
*Francesco Belardinelli, Alessio Lomuscio, and Fabio Patrizi*
## XaaS Computing

Profiling-as-a-Service: Adaptive Scalable Resource Profiling for the Cloud in the Cloud ......................................................... 157  
*Nima Kaviani, Eric Wohlstadter, and Rodger Lea*

VM Placement in non-Homogeneous IaaS-Clouds ......................... 172  
*Konstantinos Tsakalozos, Mema Roussopoulos, and Alex Delis*

## Service Discovery

Place Semantics into Context: Service Community Discovery from the WSDL Corpus......................................................... 188  
*Qi Yu*

WTCluster: Utilizing Tags for Web Services Clustering ................. 204  
*Liang Chen, Liukai Hu, Zibin Zheng, Jian Wu, Jianwei Yin, Ying Li, and Shuiguang Deng*

Similarity Function Recommender Service Using Incremental User Knowledge Acquisition ...................................................... 219  
*Seung Hwan Ryu, Boualem Benatallah, Hye-Young Paik, Yang Sok Kim, and Paul Compton*

Revealing Hidden Relations among Web Services Using Business Process Knowledge ......................................................... 235  
*Ahmed Awad and Mohammed AbuJarour*

## Service Science and Management

Towards a Service System Ontology for Service Science ............ 250  
*Elisah Lemey and Geert Poels*

Support for the Business Motivation Model in the WS-Policy4MASC Language and MiniZnMASC Middleware ............................. 265  
*Qinghua Lu, Vladimir Tosic, and Paul L. Bannerman*

WS-Governance: A Policy Language for SOA Governance ........... 280  
*José Antonio Parejo, Pablo Fernandez, and Antonio Ruiz-Cortés*

QoS-Based Task Scheduling in Crowdsourcing Environments ....... 297  
*Roman Khazankin, Harald Psaier, Daniel Schall, and Schahram Dustdar*

## Service Security and Trust

Model Driven Security Analysis of IDaaS Protocols ...................... 312  
*Apurva Kumar*
Credibility-Based Trust Management for Services in Cloud Environments .................................................. 328
   Talal H. Noor and Quan Z. Sheng

Service Monitoring
Monere: Monitoring of Service Compositions for Failure Diagnosis ......................................................... 344
   Bruno Wassermann and Wolfgang Emmerich
Multi-layered Monitoring and Adaptation .................................... 359
   Sam Guinea, Gabor Kecskemeti, Annapaola Marconi, and Branimir Wetzstein

Service Composition
Efficient, Interactive Recommendation of Mashup Composition Knowledge .................................................. 374
   Soudip Roy Chowdhury, Florian Daniel, and Fabio Casati
A Semantic and Information Retrieval Based Approach to Service Contract Selection ........................................ 389
   Silvia Calegari, Marco Comerio, Andrea Maurino, Emanuele Panzeri, and Gabriella Pasi
Modeling and Managing Variability in Process-Based Service Compositions ................................................. 404
   Tuan Nguyen, Alan Colman, and Jun Han

Quality of Service 2
QoS-Driven Proactive Adaptation of Service Composition ............. 421
   Rafael Aschoff and Andrea Zisman
A Quality Aggregation Model for Service-Oriented Software Product Lines Based on Variability and Composition Patterns ............. 436
   Bardia Mohabbati, Dragan Gašević, Marek Hatala, Mohsen Asadi, Ebrahim Bagheri, and Marko Bošković
Optimization of Complex QoS-Aware Service Compositions .......... 452
   Dieter Schuller, Artem Polyvyanyy, Luciano García-Bañuelos, and Stefan Schulte
## Table of Contents

### Research Papers – Short

#### Business Process Modeling

<table>
<thead>
<tr>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goal-Driven Business Process Derivation</td>
<td>467</td>
</tr>
<tr>
<td>Aditya K. Ghose, Nanjagud C. Narendra, Karthikeyan Ponnalagu, Anurag Panda, and Atul Gohad</td>
<td></td>
</tr>
<tr>
<td>Defining and Analysing Resource Assignments in Business Processes with RAL</td>
<td>477</td>
</tr>
<tr>
<td>Cristina Cabanillas, Manuel Resinas, and Antonio Ruiz-Cortés</td>
<td></td>
</tr>
<tr>
<td>L.A. Prashanth, H.L. Prasad, Nirmit Desai, Shalabh Bhatnagar, and Gargi Dasgupta</td>
<td></td>
</tr>
<tr>
<td>Declarative Enhancement Framework for Business Processes</td>
<td>495</td>
</tr>
<tr>
<td>Heerko Groefsema, Pavel Bulanov, and Marco Aiello</td>
<td></td>
</tr>
</tbody>
</table>

#### XaaS Computing

<table>
<thead>
<tr>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>RSCMap: Resiliency Planning in Storage Clouds</td>
<td>505</td>
</tr>
<tr>
<td>Vimmi Jaiswal, Aritra Sen, and Akshat Verma</td>
<td></td>
</tr>
<tr>
<td>Dynamically Selecting Composition Algorithms for Economical Composition as a Service</td>
<td>513</td>
</tr>
<tr>
<td>Immanuel Trummer and Boi Faltings</td>
<td></td>
</tr>
<tr>
<td>A Service Model for Development and Test Clouds</td>
<td>523</td>
</tr>
<tr>
<td>Debdoot Mukherjee, Monika Gupta, Vibha Singhal Sinha, and Nianjun Zhou</td>
<td></td>
</tr>
</tbody>
</table>

#### Quality of Service

<table>
<thead>
<tr>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time Based QoS Modeling and Prediction for Web Services</td>
<td>532</td>
</tr>
<tr>
<td>Leilei Chen, Jian Yang, and Liang Zhang</td>
<td></td>
</tr>
<tr>
<td>CANPRO: A Conflict-Aware Protocol for Negotiation of Cloud Resources and Services</td>
<td>541</td>
</tr>
<tr>
<td>Marco A.S. Netto</td>
<td></td>
</tr>
<tr>
<td>Game-Theoretic Analysis of a Web Services Collaborative Mechanism</td>
<td>549</td>
</tr>
<tr>
<td>Babak Khosravifar, Jamal Bentahar, Kathleen Clacens, Christophe Goffart, and Philippe Thiran</td>
<td></td>
</tr>
<tr>
<td>Title</td>
<td>Page</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Importance Sampling of Probabilistic Contracts in Web Services</td>
<td>557</td>
</tr>
<tr>
<td><em>Ajay Kattepur</em></td>
<td></td>
</tr>
<tr>
<td>Particle Filtering Based Availability Prediction for Web Services</td>
<td>566</td>
</tr>
<tr>
<td><em>Lina Yao and Quan Z. Sheng</em></td>
<td></td>
</tr>
<tr>
<td>A Penalty-Based Approach for QoS Dissatisfaction Using Fuzzy Rules</td>
<td>574</td>
</tr>
<tr>
<td><em>Barbara Pernici, S. Hossein Siadat, Salima Benbernou, and Mourad Ouziri</em></td>
<td></td>
</tr>
</tbody>
</table>

**Service Runtime Infrastructures**

<table>
<thead>
<tr>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cellular Differentiation-Based Service Adaptation</td>
<td>582</td>
</tr>
<tr>
<td><em>Ichiro Satoh</em></td>
<td></td>
</tr>
<tr>
<td>Graceful Interruption of Request-Response Service Interactions</td>
<td>590</td>
</tr>
<tr>
<td><em>Mila Dalla Preda, Maurizio Gabrieelli, Ivan Lanese, Jacopo Mauro, and Gianluigi Zavattaro</em></td>
<td></td>
</tr>
<tr>
<td>Adaptation of Web Service Interactions Using Complex Event Processing Patterns</td>
<td>601</td>
</tr>
<tr>
<td><em>Yehia Taher, Michael Parkin, Mike P. Papazoglou, and Willem-Jan van den Heuvel</em></td>
<td></td>
</tr>
</tbody>
</table>

**Service Migration and Adoption**

<table>
<thead>
<tr>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Employing Dynamic Object Offloading as a Design Breakthrough for SOA Adoption</td>
<td>610</td>
</tr>
<tr>
<td><em>Quirino Zagarese, Gerardo Canfora, and Eugenio Zimeo</em></td>
<td></td>
</tr>
<tr>
<td>A Survey of SOA Migration in Industry</td>
<td>618</td>
</tr>
<tr>
<td><em>Maryam Razavian and Patricia Lago</em></td>
<td></td>
</tr>
</tbody>
</table>

**Service Composition**

<table>
<thead>
<tr>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Forms-Based Service Composition</td>
<td>627</td>
</tr>
<tr>
<td><em>Ingo Weber, Hye-Young Paik, and Boualem Benatallah</em></td>
<td></td>
</tr>
<tr>
<td>Contractually Compliant Service Compositions</td>
<td>636</td>
</tr>
<tr>
<td><em>Enrique Martinez, Gregorio Diaz, and M. Emilia Cambroner</em></td>
<td></td>
</tr>
<tr>
<td>Profit Sharing in Service Composition</td>
<td>645</td>
</tr>
<tr>
<td><em>Shigeo Matsubara</em></td>
<td></td>
</tr>
</tbody>
</table>
Table of Contents

Service Applications

A Predictive Business Agility Model for Service Oriented Architectures .................................................. 653
    Mamoun Hirzalla, Peter Bahrs, Jane Cleland-Huang, Craig S. Miller, and Rob High

Personal-Hosting RESTful Web Services for Social Network Based Recommendation ........................................... 661
    Youliang Zhong, Weiliang Zhao, and Jian Yang

Work as a Service ........................................ 669
    Daniel V. Oppenheimer, Lav R. Varshney, and Yi-Min Chee

Author Index ............................................................................. 679
Contractually Compliant Service Compositions

Enrique Martínez, Gregorio Díaz, and M. Emilia Cambronero
Department of Computer Science
University of Castilla - La Mancha, Spain
{emartinez,gregorio,emicp}@dsi.uclm.es

Abstract. In the field of service-oriented computing, an e-contract is used to regulate the acceptable behaviours of the services taking part in a composition. C-O Diagrams are a visual model for the specification of deontic e-contracts, including reparations, conditional clauses and real-time restrictions. In this work we define a set of satisfaction rules based on timed automata to see whether a composition is compliant with the contract specification, providing the model with the mathematical rigour necessary for formal verification.

Keywords: Contracts, deontic logic, formal verification, visual models, timed automata.

1 Introduction

An e-contract is defined as a contract regulating business-to-business interactions over the Internet. Its purpose is to ensure that the partners taking part in the business process (possibly Web Services) comply with certain obligations, permissions and prohibitions by means of specifying a set of clauses. Moreover, these clauses can include the conditions required to be applied, the time boundaries to be satisfied, and references to secondary contracts (reparations).

In [4] we have already presented a visual model for the specification of e-contracts called C-O Diagrams. The approach followed is inspired by the formal language $\mathcal{CL}$ [5], in which a contract is expressed as a composition of obligations, permissions and prohibitions over actions, and reparations for obligations and prohibitions can be defined. The main contribution of this work is the definition of a set of satisfaction rules for C-O Diagrams based on timed automata [1]. These rules allow us to formally check if all the possible behaviours of a service composition are compliant with the e-contract specified by a C-O Diagram. Let us note that in this work we does not focus on how the behaviours of a service composition are translated into timed automata, as many other works in the literature are related to this aspect (e.g. [2,3]).

The rest of the work is structured as follows: Section 2 is a brief description of C-O Diagrams and its syntax. Section 3 defines the set of satisfaction rules. Finally, in Section 4 we present the conclusions and future work.

* Partially supported by the Spanish government (cofinanced by FEDER founds) with the project TIN2009-14312-C02-02 and the JCCLM regional project PEII09-023-7745. The first author is supported by the European Social Fund and the JCCLM.

2 C-O Diagrams Description and Syntax

In Fig. 1 we show the basic element of C-O Diagrams. It is called a box and it is divided into four fields. On the left-hand side of the box we specify the conditions and restrictions. The guard g specifies the conditions under which the contract clause must be taken into account (boolean expression). The time restriction tr specifies the time frame during which the contract clause must be satisfied (deadlines, timeouts, etc.). The propositional content P, on the center, is the main field of the box, and it is used to specify normative aspects (obligations, permissions and prohibitions) that are applied over actions, and/or the specification of the actions themselves. The last field of these boxes, on the right-hand side, is the reparation R. This reparation, if specified by the contract clause, is a reference to another contract that must be satisfied in case the main norm is not satisfied (a prohibition is violated or an obligation is not fulfilled, there is no reparation for permission), considering the clause eventually satisfied if this reparation is satisfied. Each box has also a name and an agent. The name is useful both to describe the clause and to reference the box from other clauses, so it must be unique. The agent indicates who is the performer of the action.

These basic elements of C-O Diagrams can be refined by using AND/OR/SEQ refinements, as shown in Fig. 2. The aim of these refinements is to capture the hierarchical clause structure followed by most contracts. An AND-refinement means that all the subclauses must be satisfied in order to satisfy the parent clause. An OR-refinement means that it is only necessary to satisfy one of the subclauses in order to satisfy the parent clause, so as soon as one of its subclauses is fulfilled, we conclude that the parent clause is fulfilled as well. A SEQ-refinement means that the norm specified in the target box (SubClause2 in Fig. 2) must be fulfilled after satisfying the norm specified in the source box (SubClause1 in Fig. 2). By using these structures we can build a hierarchical tree with the clauses defined by a contract, where the leaf clauses correspond to the atomic clauses, that is, to the clauses that cannot be divided into subclauses. There is another structure that can be used to model repetition. This structure

![Fig. 1. Box structure](image)

![Fig. 2. AND/OR/SEQ refinements and repetition in C-O Diagrams](image)
is represented as an arrow going from a subclause to one of its ancestor clauses (or to itself), meaning the repetitive application of all the subclauses of the target clause after satisfying the source subclause. For example, in the right-hand side of Fig. 2 we have an OR-refinement with an arrow going from SubClause1 to SubClause2. It means that after satisfying SubClause1 we apply Clause again, but not after satisfying SubClause2.

We have given here an abridged description of C-O Diagrams. A more detailed description can be found in [4], including a qualitative and quantitative evaluation underlining the advantages of having a visual model for the specification of e-contracts, and a discussion on related work.

**Definition 1. (C-O Diagrams Syntax)** We consider a finite set of real-valued variables $C$ standing for clocks, a finite set of non-negative integer-valued variables $V$, a finite alphabet $\Sigma$ for atomic actions, a finite set of identifiers $A$ for agents, and another finite set of identifiers $N$ for names. The greek letter $\epsilon$ means that and expression is not given, i.e., it is empty.

We use $C$ to denote the contract modelled by a C-O Diagram. The diagram is defined by the following EBNF grammar:

$$
C := (\text{agent, name, } g, \text{tr, } O(C_2), R) | (\text{agent, name, } g, \text{tr, } P(C_2), \epsilon) | (\text{agent, name, } g, \text{tr, } F(C_2), R) | (\epsilon, \text{name, } g, \text{tr, } C_1, \epsilon) \\
C_1 := C (\text{And } C)^+ | C (\text{Or } C)^+ | C (\text{Seq } C)^+ \\
C_2 := a | C_3 (\text{And } C_3)^+ | C_3 (\text{Or } C_3)^+ | C_3 (\text{Seq } C_3)^+ \\
C_3 := (\epsilon, \text{name, } \epsilon, \epsilon, C_2, \epsilon) \\
R := C | \epsilon
$$

where $a \in \Sigma$, agent $\in A$ and name $\in N$. Guard $g$ is $\epsilon$ or a conjunctive formula of atomic constraints of the form: $v \sim n$ or $v - w \sim n$, for $v, w \in V$, $\sim \in \{=,<,\leq,\geq\}$ and $n \in \mathbb{N}$, whereas timed restriction $\epsilon$ is $\epsilon$ or a conjunctive formula of atomic constraints of the form: $x \sim n$ or $x - y \sim n$, for $x, y \in C$, $\sim \in \{=,<,\leq,\geq\}$ and $n \in \mathbb{N}$. $O$, $P$ and $F$ are the deontic operators corresponding to obligation, permission and prohibition, respectively, where $O(C_2)$ states the obligation of performing $C_2$, $F(C_2)$ states prohibition of performing $C_2$, and $P(C_2)$ states the permission of performing $C_2$. And, Or and Seq are the operators corresponding to the refinements we have in C-O Diagrams, AND-refinement, OR-refinement and SEQ-refinement, respectively.

The simplest contract we can have in C-O Diagrams is that composed of only one box including the elements agent and name. Optionally, we can specify a guard $g$ and a time restriction $\text{tr}$. We also have a deontic operator ($O$, $P$ or $F$) applied over an atomic action $a$, and in the case of obligations and prohibitions it is possible to specify another contract $C$ as a reparation.

We use $C_1$ to define a more complex contract where we combine different deontic norms by means of any of the different refinements we have in C-O Diagrams. In the box where we have the refinement into $C_1$ we cannot specify an agent nor a reparation because these elements are always related to a single deontic norm, but we still can specify a guard $g$ and a time restriction $\text{tr}$ that affect all the deontic norms we combine.
3 C-O Diagrams Satisfaction Rules

In this section we define a set of satisfaction rules for C-O Diagrams based on timed automata. The purpose of this definition is providing a formal mechanism to check if a service composition behaviour (specified by a timed automaton) is compliant with respect to a contract (specified by a C-O Diagram). To define this satisfaction rules we follow the C-O Diagrams syntax given in Definition 1. The satisfiability of a contract is defined based on the states of a timed labelled transition system associated to a timed automaton. Basically, a timed automaton (TA) \([1]\) is a tuple \((N, n_0, E, I)\), where \(N\) is a finite set of locations (nodes), \(n_0 \in N\) is the initial location, \(E\) is the set of edges, and \(I\) is a function that assigns invariant conditions (which could be empty) to locations. We write \(n \xrightarrow{g,a,r} n'\) to denote \((n, g, a, s, r, n') \in E\), where \(n, n' \in N\), \(g\) is a guard, \(a\) is an action, \(r\) is a set of clocks we want to reset, and \(s\) is a set of variable assignments. The semantics of a timed automaton is defined as a timed labelled transition system \((Q, q_0, \rightarrow)\), where \(Q\) is a set of states, \(q_0 \in Q\) is the initial state, and \(\rightarrow\) is the set of transitions. Due to the lack of space, refer to [2] for a complete definition of timed automaton and its semantics.

The C-O Diagrams satisfaction rules consist of a set of rules where the satisfiability of a contract is defined based on the states of the timed labelled transition system associated to a timed automaton. To define this set of rules we follow the C-O Diagrams syntax given in Definition 1.

Definition 2. (C-O Diagrams Satisfaction Rules: Part I)

Let \(A = (N, n_0, E, I)\) be a timed automaton, with the associated timed labelled transition system \((Q, q_0, \rightarrow)\) and \(q \in Q\). Given a C-O Diagram \(C\), one can define \((A, q) \models C\) (\(A\) in state \(q\) satisfies contract \(C\)) as follows:

1. \((A, q) \models (agent, name, g, tr, O(a), R)\) if \(\forall (q_1 \rightarrow q_2 \rightarrow \ldots \rightarrow q_j)\) for \(q = q_1:\)
   - The main clause holds, that is, \(\exists i \in [1, j - 1]\) such that \(q_i \xrightarrow{a} q_{i+1}\) with \(n_i \xrightarrow{g_i,a,r} n_{i+1}\) where \((g \land tr) \in gt\) and \(agent(a)\)
   - The main clause does not hold but reparation holds, that is, \(R \neq \epsilon\) and \((A, q_{i+1}) \models R\) for the first \(i \in [1, j - 1]\) such that \(q_i \xrightarrow{d} q_{i+1}\) with \((n_i, u) \xrightarrow{d} (n_i, u + d)\) and \((u + d) \notin tr\)

2. \((A, q) \models (agent, name, g, tr, P(a), \epsilon)\) if \(\exists (q_1 \rightarrow q_2 \rightarrow \ldots \rightarrow q_j)\) for \(q = q_1\) where the main clause holds, that is, \(\exists i \in [1, j - 1]\) such that \(q_i \xrightarrow{a} q_{i+1}\) with \(n_i \xrightarrow{g_i,a,r} n_{i+1}\) where \((g \land tr) \in gt\) and \(agent(a)\)
Lines (1)–(3) correspond to the satisfaction rules of applying an obligation, a permission or a prohibition over an atomic action \( a \). In the case of obligation, for all the possible paths in our automaton we must have the performance of \( a \) by the specified agent (denoted by \( \text{agent}(a) \)) and fulfilling also any condition or time restriction we have specified (denoted by \( (g \land tr) \in gt \)). If the obliged action is not performed in the expected time frame, we have the alternative possibility of satisfying reparation \( R \) from the moment at which timed restriction is not fulfilled anymore (denoted by \((u + d) \notin tr\)). In permission we consider that the performance of \( a \) is only necessary in one of the paths. This interpretation of permission is because we think that an automaton satisfying a contract must offer the possibility of performing a permitted action in at least one of its paths. Prohibition is the opposite of permission, so we cannot have a path where we perform the forbidden action \( a \), but in case we perform the action we still have the possibility of satisfying reparation \( R \) after that.

**Definition 2.** (C-O Diagrams Satisfaction rules: Part II)

\( (A, q) \models (\text{agent}, \text{name}, g, tr, F(a), R) \) iff \( \forall (q_1 \rightarrow q_2 \rightarrow \ldots \rightarrow q_i) \) for \( q = q_i \):
- The **main clause holds**, that is, \( \exists i \in \{1, j - 1\} \) such that \( q_i \xrightarrow{a} q_{i+1} \) with \( n_i \xrightarrow{g \land tr} n_{i+1} \) where \((g \land tr) \in gt\) and \( \text{agent}(a) \).
- The **main clause does not hold** but **reparation holds**, that is, \( R \neq \epsilon \) and \( (A, q_{i+1}) \models R \) for the first \( i \in \{1, j - 1\} \) such that \( q_i \xrightarrow{a} q_{i+1} \) with \( n_i \xrightarrow{g \land tr} n_{i+1} \) where \((g \land tr) \in gt\) and \( \text{agent}(a) \).

\[ (3) \text{(A, q)} \models (\text{agent}, \text{name}, g, tr, F(a), R) \text{iff} \forall (q_1 \rightarrow q_2 \rightarrow \ldots \rightarrow q_i) \text{ for } q = q_i: \]
- \( \text{The main clause holds}, \) that is, \( \exists i \in \{1, j - 1\} \) such that \( q_i \xrightarrow{a} q_{i+1} \) with \( n_i \xrightarrow{g \land tr} n_{i+1} \) where \((g \land tr) \in gt\) and \( \text{agent}(a) \).
- \( \text{The main clause does not hold but reparation holds}, \) that is, \( R \neq \epsilon \) and \( (A, q_{i+1}) \models R \) for the first \( i \in \{1, j - 1\} \) such that \( q_i \xrightarrow{a} q_{i+1} \) with \( n_i \xrightarrow{g \land tr} n_{i+1} \) where \((g \land tr) \in gt\) and \( \text{agent}(a) \).
Definition 2. (C-O Diagrams Satisfaction Rules: Part III)

(7) \( (A, q) \rightarrow (agent, name, g, tr, D((\epsilon, name_1, g_1, tr_1, C_1, \epsilon) \ \text{REF} (\epsilon, name_2, g_2, tr_2, C_2, \epsilon) \ \text{REF} \ldots \text{REF} (\epsilon, name_k, g_k, tr_k, C_k, \epsilon)), R) \iff (A, q) \rightarrow (\epsilon, name, g, tr, (agent_1, name_1, g_1, tr_1, C_1, R_1) \ \text{REF} (agent_2, name_2, g_2, tr_2, C_2, R_2) \ \text{REF} \ldots \text{REF} (agent_k, name_k, g_k, tr_k, C_k, R_k)) \)

(8) \( (A, q) \rightarrow (\epsilon, name, g, tr, (agent_1, name_1, g_1, tr_1, C_1, R_1) \ \text{Seq} (agent_2, name_2, g_2, tr_2, C_2, R_2) \ \text{Seq} \ldots \text{Seq} (agent_k, name_k, g_k, tr_k, C_k, R_k)) \)

(9) \( (A, q) \rightarrow (\epsilon, name, g, tr, (agent_1, name_1, g_1, tr_1, D(C_1), (\epsilon, name_{m_1}, g_{m_1}, tr_{m_1}, C_{m_1}, \epsilon)), R_1) \ \text{Seq} (agent_2, name_2, g_2, tr_2, (\epsilon, name_{m_2}, g_{m_2}, tr_{m_2}, C_{m_2}, \epsilon)), R_2) \ \text{Seq} \ldots \text{Seq} (agent_k, name_k, g_k, tr_k, C_k, R_k), \epsilon) \)

In line (7) we have that \( D \in \{O, P, F\} \) and \( \text{REF} \in \{\text{And, Or, Seq}\} \), so it corresponds to the satisfaction rule of applying a deontic norm over an AND-refinement, an OR-refinement or a SEQ-refinement of subcontracts. For all the deontic operators we just propagate them into each one of the subcontracts, as well as reparation \( R \) and \( agent \), and the satisfaction of the main contract consists of the satisfaction of the refinement \( \text{REF} \) of these new subcontracts.

In line (8) we have that \( \text{REF} \in \{\text{And, OR}\} \) and \( \diamond \in \{\land, \lor\} \). It corresponds to the satisfaction rule for an AND-refinement or an OR-refinement of subcontracts with no deontic operator applied over the refinements (they will be specified in the subcontracts). In these cases, the satisfaction of the main contract consists of the conjunction (\( \land \) for AND-refinement) or disjunction (\( \lor \) for OR-refinement) of the satisfaction of each one of the subcontracts, propagating any condition or time restriction in the main contract into these subcontracts (denoted by \( g \land g_k \) and \( tr \land tr_k \)).

Line (9) corresponds to the satisfaction rule for a SEQ-refinement when the first element of the sequence is a deontic norm \( D \in \{O, P, F\} \) applied over another refinement of subcontracts \( \text{REF} \in \{\text{And, Or, Seq}\} \). In all these cases we propagate the deontic operator into each one of the subcontracts, as well as
reparation $R$ and $agent$, so the satisfaction of the main contract consists of the satisfaction of the SEQ-refinement when the first element of the sequence is the refinement $RE$. We have before but now combining these new subcontracts.

**Definition 2.** (C-O Diagrams) Satisfaction Rules: Part IV

1. $(A, q) \models (\epsilon, name, g, tr, (\epsilon, name_1, g_1, tr_1, (agent_{11}, name_{11}, g_{11}, tr_{11}, C_{11}, R_{11}) \rightarrow (agent_{12}, name_{12}, g_{12}, tr_{12}, C_{12}, R_{12}) \rightarrow \ldots \rightarrow RE (agent_{i1}, name_{i1}, g_{i1}, tr_{i1}, C_{i1}, R_{i1}), \epsilon) \rightarrow Seq (agent_{i2}, name_{i2}, g_{i2}, tr_{i2}, C_{i2}, R_{i2}) \rightarrow \ldots \rightarrow RE \rightarrow (agent_{k1}, name_{k1}, g_{k1}, tr_{k1}, C_{k1}, R_{k1}), \epsilon)$

2. $(A, q) \models (\epsilon, name, g, tr, (agent_{11}, name_{11}, g_{11} \land g_{11} \land g_{12} \land tr_{11} \land tr_{12}, C_{11}, R_{11}) \rightarrow Seq (agent_{k1}, name_{k1}, g_{k1}, tr_{k1}, C_{k1}, R_{k1}), \epsilon)$

Lines (10) corresponds to the satisfaction rule for a SEQ-refinement when the first element of the sequence is a refinement $RE \in \{And, OR\}$ of subcontracts, with no deontic operator applied over it. We consider SEQ-refinement as distributive over AND-refinement and OR-refinement, so the satisfaction of this contract consists of the satisfaction of the contract where we have applied this property, propagating any condition or time restriction in the AND-refinement or the OR-refinement into their subcontracts.

Finally, line (11) corresponds to the satisfaction rule for a SEQ-refinement when the first element of the sequence is another SEQ-refinement of subcontracts, with no deontic operator applied over it. We consider SEQ-refinement to be associative, so the satisfaction of this contract consists of the satisfaction of the contract where we have applied this property to have only one SEQ-refinement, propagating any condition or time restriction in the internal SEQ-refinement into their subcontracts.

After defining these satisfaction rules, the definition of the algorithm checking whether a timed automaton $A = (N, n_0, E, I)$ with associated timed labelled transition system $(Q, q_0, \rightarrow)$ satisfies a C-O Diagram $C$ is quite straightforward. It returns “YES” if $(A, q_0) \models C$, otherwise it returns “NO”.

**Example 1.** Let us consider the Second Update C-O Diagram of the Software Provision System case study presented in [4]. It models a contract we denote as $C$. According to the C-O Diagrams syntax, this diagram can be written as:

$(\epsilon, Second\_Update, \epsilon, \epsilon, (Software\_provider, Sends\_Update2, \epsilon, tr_4, O(a_4), R_4)) \rightarrow \ldots \rightarrow Seq (\epsilon, Client\_Second\_Behavior, \epsilon, \epsilon, (Client, Second\_Payment, \epsilon, \epsilon, O(a_5), \epsilon)) \rightarrow \ldots \rightarrow And (Client, Second\_Changes, \epsilon, \epsilon, F(a_6), R_6, \epsilon)$
where we use $tr_4$ to denote the temporal restriction we have in $Sends_Update2$, $R_4$ is the reparation we define for this clause, and $R_6$ is the reparation we define for clause $Second_Changes$, stating the obligation of performing $r_{6a}$ or $r_{6b}$.

![Fig. 3. Automaton $A$ of Example 1](image)

We want to check if the timed automaton $A$ shown in Fig. 3 satisfies this contract $C$ starting from $n_0$. First, we have a SEQ-refinement where the first subcontract states the obligation of performing $a_4$. Then, we apply rule (4) and we see that the initial obligation is fulfilled, performing $Software\ provider\ a_4$ within time frame $tr_4$ in the edge from $n_0$ to $n_1$. Next, we have an AND-refinement between an obligation and a prohibition, so applying rule (8) we have to check the satisfaction of the obligation and the satisfaction of the prohibition from $n_1$. According to rule (1), we have to see that $a_5$ is performed by $Client$ in every possible path. An edge performing $a_5$ exists in all cases, so the obligation is satisfied. According to rule (3), we have to see that $a_6$ is not performed by $Client$ in any possible path. For path $n_1 \rightarrow n_2$ it is fulfilled, but for path $n_1 \rightarrow n_3$ we have that $a_6$ is performed. Nevertheless, as there is a reparation $R_6$ defined, we have to see if this reparation is fulfilled. We can see that, after applying rules (7) and (8), it is enough to satisfy the obligation of one of the two actions ($r_{6a}$ or $r_{6b}$) in order to fulfill the reparation. We have that the edge from $n_4$ to $n_5$ performs $r_{6a}$, so the reparation is fulfilled and the subcontract is eventually fulfilled. Therefore, we conclude that contract $C$ is satisfied by $A$.

4 Conclusions and Future Work

In this paper we have defined a set of satisfaction rules for $C-O\ Diagrams$ based on timed automata to check whether a service composition is compliant with a contract specified by a $C-O\ Diagram$. We have also shown an example about how to apply this approach.

As future work, we are planning to apply this model to several case studies in order to check its usefulness in different fields and the evaluation of its computational complexity. We are also working on the definition of a transformation that automatically generates a timed automaton compliant with the contract specified by a $C-O\ Diagram$. This can be useful for several purposes, for example to check the correctness of the contract specification.
References