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 PEII09-0232-7745. The first author is supported by the European Social Fund and the JCCLM.

G. Kappel, Z. Maamar, H.R. Motahari-Nezhad (Eds.): ICSOC 2011, LNCS 7084, pp. 636–644, 2011.

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.

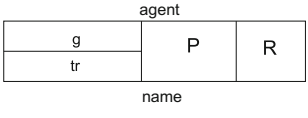

Fig. 1. Box structure

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 satisfied 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

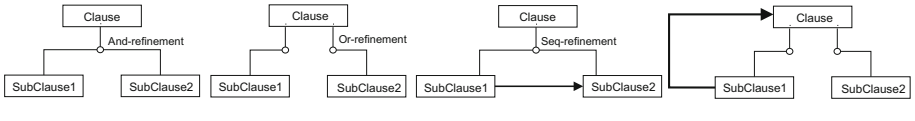

Fig. 2. AND/OR/SEQ refinements and repetition in *C-O Diagrams*

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 *Clause*. 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 detail 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 $\mathcal{C}$ standing for clocks, a finite set of non-negative integer-valued variables $\mathcal{V}$, a finite alphabet Σ for atomic actions, a finite set of identifiers $\mathcal{A}$ for agents, and another finite set of identifiers $\mathcal{N}$ for names. The greek letter ϵ 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:

$$
\begin{aligned}
C &:= (agent, name, g, tr, O(C_2), R) \mid \\
&\quad (agent, name, g, tr, P(C_2), \epsilon) \mid \\
&\quad (agent, name, g, tr, F(C_2), R) \mid \\
&\quad (\epsilon, name, g, tr, C_1, \epsilon) \\
C_1 &:= C\,(And\ C)^+ \mid C\,(Or\ C)^+ \mid C\,(Seq\ C)^+ \\
C_2 &:= a \mid C_3\,(And\ C_3)^+ \mid C_3\,(Or\ C_3)^+ \mid C_3\,(Seq\ C_3)^+ \\
C_3 &:= (\epsilon, name, \epsilon, \epsilon, C_2, \epsilon) \\
R &:= C \mid \epsilon
\end{aligned}
$$

where $a \in \Sigma$, $agent \in \mathcal{A}$ and $name \in \mathcal{N}$. Guard g is ϵ or a conjunctive formula of atomic constraints of the form: $v \sim n$ or $v - w \sim n$, for $v, w \in \mathcal{V}$, $\sim \in \{\leq, <, =, >, \geq\}$ and $n \in \mathbb{N}$, whereas timed restriction tr is ϵ or a conjunctive formula of atomic constraints of the form: $x \sim n$ or $x - y \sim n$, for $x, y \in \mathcal{C}$, $\sim \in \{\leq, <, =, >, \geq\}$ 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 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 tr that affect all the deontic norms we combine.

Once we write a deontic operator in a box of our diagram, we have two possibilities as we can see in the specification of C_2: we can just write a simple action a in the box, being the deontic operator applied only over it, or we can refine this box in order to apply the deontic operator over a compound action. In this case we have that the subboxes (C_3) cannot define a new deontic operator as it has already been defined in the parent box (affecting all the subboxes).

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 $n \xrightarrow{g,a,r}_{s} n'$ 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 $\mathcal{A} = (N, n_0, E, I)$ 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 $(\mathcal{A}, q) \models C$ ($\mathcal{A}$ in state q satisfies contract C) as follows:

(1) $(\mathcal{A}, q) \models (agent, name, g, tr, O(a), R)$ **iff** $\forall \langle q_1 \longrightarrow q_2 \longrightarrow \ldots \longrightarrow q_j \rangle$ for $q = q_1$:
- The **main clause holds**, that is, $\exists i \in [1, j-1]$ such that $q_i \xrightarrow{a}_{s} q_{i+1}$ with $n_i \xrightarrow{g\prime,a,r}_{s} n_{i+1}$ where $(g \wedge tr) \in g\prime$ and $agent(a)$
- The **main clause does not hold** but **reparation holds**, that is, $R \neq \epsilon$ and $(\mathcal{A}, q_{i+1}) \models R$ for the first $i \in [1, j-1]$ such that $q_i \xrightarrow{d}_{s} q_{i+1}$ with $(n_i, u) \xrightarrow{d} (n_i, u + d)$ and $(u + d) \notin tr$

(2) $(\mathcal{A}, q) \models (agent, name, g, tr, P(a), \epsilon)$ **iff** $\exists \langle q_1 \longrightarrow q_2 \longrightarrow \ldots \longrightarrow q_j \rangle$ for $q = q_1$ where **the main clause holds**, that is, $\exists i \in [1, j-1]$ such that $q_i \xrightarrow{a}_{s} q_{i+1}$ with $n_i \xrightarrow{g\prime,a,r}_{s} n_{i+1}$ where $(g \wedge tr) \in g\prime$ and $agent(a)$

(3) $(\mathcal{A}, q) \models (agent, name, g, tr, F(a), R)$ **iff** $\forall \langle q_1 \longrightarrow q_2 \longrightarrow \ldots \longrightarrow q_j \rangle$ for $q = q_1$:
– The **main clause holds**, that is, $\nexists i \in [1, j-1]$ such that $q_i \xrightarrow[s]{a} q_{i+1}$ with $n_i \xrightarrow[s]{g\prime,a,r} n_{i+1}$ where $(g \wedge tr) \in g\prime$ and $agent(a)$,
– The **main clause does not hold** but **reparation holds**, that is, $R \neq \epsilon$ and $(\mathcal{A}, q_{i+1}) \models R$ for the first $i \in [1, j-1]$ such that $q_i \xrightarrow[s]{a} q_{i+1}$ with $n_i \xrightarrow[s]{g\prime,a,r} n_{i+1}$ where $(g \wedge tr) \in g\prime$ 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 $agent(a)$) and fulfilling also any condition or time restriction we have specified (denoted by $(g \wedge tr) \in g\prime$). 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)

(4) $(\mathcal{A}, q) \models (\epsilon, name, g, tr, (agent_1, name_1, g_1, tr_1, O(a), R_1)\, Seq$
$(agent_2, name_2, g_2, tr_2, C_2, R_2) Seq \ldots Seq\, (agent_k, name_k, g_k, tr_k, C_k, R_k), \epsilon)$
iff $\forall \langle q_1 \longrightarrow q_2 \longrightarrow \ldots \longrightarrow q_j \rangle$ for $q = q_1$:
– The **first main clause holds**, that is, $\exists i \in [1, j-1]$ such that $q_i \xrightarrow[s]{a} q_{i+1}$ with $n_i \xrightarrow[s]{g\prime,a,r} n_{i+1}$ where $(g \wedge tr) \in g\prime$ and $agent(a)$, **and remaining sequence holds**, that is,
$(\mathcal{A}, q_{i+1}) \models (\epsilon, name, g, tr, (agent_2, name_2, g_2, tr_2, C_2, R_2)\, Seq$
$\ldots Seq\ (agent_k, name_k, g_k, tr_k, C_k, R_k), \epsilon)$
– The **first main clause does not hold** but its **reparation and remaining sequence holds**, that is, $R \neq \epsilon$ and
$(\mathcal{A}, q_{i+1}) \models (\epsilon, name, g, tr, R_1\, Seq\, (agent_2, name_2, g_2, tr_2, C_2, R_2) Seq \ldots Seq$
$(agent_k, name_k, g_k, tr_k, C_k, R_k), \epsilon)$ for the first $i \in [1, j-1]$ such that $q_i \xrightarrow[s]{d} q_{i+1}$
with $(n_i, u) \xrightarrow{d} (n_i, u + d)$ and $(u + d) \notin tr$

(5) $(\mathcal{A}, q) \models (\epsilon, name, g, tr, (agent_1, name_1, g_1, tr_1, P(a), R_1)\, Seq$
$(agent_2, name_2, g_2, tr_2, C_2, R_2) Seq \ldots Seq\, (agent_k, name_k, g_k, tr_k, C_k, R_k), \epsilon)$
iff $\exists \langle q_1 \longrightarrow q_2 \longrightarrow \ldots \longrightarrow q_j \rangle$ for $q = q_1$ where the **first main clause holds**, that is, $\exists i \in [1, j-1]$ such that $q_i \xrightarrow[s]{a} q_{i+1}$ with $n_i \xrightarrow[s]{g\prime,a,r} n_{i+1}$ where
$(g \wedge tr) \in g\prime$ and $agent(a)$, and **remaining sequence holds**, that is,
$(\mathcal{A}, q_{i+1}) \models (\epsilon, name, g, tr, (agent_2, name_2, g_2, tr_2, C_2, R_2)\, Seq \ldots Seq$
$(agent_k, name_k, g_k, tr_k, C_k, R_k), \epsilon)$

(6) $(\mathcal{A}, q) \models (\epsilon, name, g, tr, (agent_1, name_1, g_1, tr_1, F(a), R_1)\, Seq$
$(agent_2, name_2, g_2, tr_2, C_2, R_2) Seq \ldots Seq\, (agent_k, name_k, g_k, tr_k, C_k, R_k), \epsilon)$
iff $\forall \langle q_1 \longrightarrow q_2 \longrightarrow \ldots \longrightarrow q_j \rangle$ for $q = q_1$:
– The **first main clause holds**, that is, $\nexists i \in [1, j-1]$ such that $q_i \xrightarrow[s]{a} q_{i+1}$ with $n_i \xrightarrow[s]{g\prime,a,r} n_{i+1}$ where $(g \wedge tr) \in g\prime$ and $agent(a)$, **and remaining sequence holds**, that is,
$(\mathcal{A}, q_{i+1}) \models (\epsilon, name, g, tr, (agent_2, name_2, g_2, tr_2, C_2, R_2)\, Seq$
$\ldots Seq\ (agent_k, name_k, g_k, tr_k, C_k, R_k), \epsilon)$
– The **first main clause does not hold** but its **reparation and remaining sequence holds**, that is, $R \neq \epsilon$ and
$(\mathcal{A}, q_{i+1}) \models (\epsilon, name, g, tr, R_1\, Seq\, (agent_2, name_2, g_2, tr_2, C_2, R_2) Seq \ldots Seq$

$(agent_k, name_k, g_k, tr_k, C_k, R_k), \epsilon)$ for the first $i \in [1, j-1]$ such that $q_i \xrightarrow{a}_s q_{i+1}$ with $n_i \xrightarrow[s]{g\prime, a, r} n_{i+1}$ where $(g \wedge tr) \in g\prime$ and $agent(a)$

Lines (4)–(6) correspond to the satisfaction rules for a contract consisting of a SEQ-refinement when the first subcontract of the sequence is just a deontic operator applied over an atomic action. The satisfaction of the contract consists of the satisfaction of the first subcontract of the sequence and, after that, the satisfaction of the rest of the sequence starting from a location we reach after satisfying the first subcontract. Alternatively, if reparation R of the first subcontract is not empty, we can satisfy a SEQ-refinement having this reparation as its first subcontract.

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

(7) $(\mathcal{A}, q) \models (agent, name, g, tr, \mathcal{D}((\epsilon, name_1, g_1, tr_1, C_1, \epsilon)\, \mathcal{REF}$
$(\epsilon, name_2, g_2, tr_2, C_2, \epsilon)\, \mathcal{REF} \ldots \mathcal{REF}\, (\epsilon, name_k, g_k, tr_k, C_k, \epsilon)), R)$ **iff**
$(\mathcal{A}, q) \models (\epsilon, name, g, tr, (agent, name_1, g_1, tr_1, \mathcal{D}(C_1), R)\, \mathcal{REF}$
$(agent, name_2, g_2, tr_2, \mathcal{D}(C_2), R)\, \mathcal{REF} \ldots \mathcal{REF}$
$(agent, name_k, g_k, tr_k, \mathcal{D}(C_k), R), \epsilon)$

(8) $(\mathcal{A}, q) \models (\epsilon, name, g, tr, (agent_1, name_1, g_1, tr_1, C_1, R_1)\, \mathcal{RE}$
$(agent_2, name_2, g_2, tr_2, C_2, R_2)\, \mathcal{RE} \ldots \mathcal{RE}$
$(agent_k, name_k, g_k, tr_k, C_k, R_k), \epsilon)$ **iff**
$(\mathcal{A}, q) \models (agent_1, name_1, g \wedge g_1, tr \wedge tr_1, C_1, R_1) \diamond$
$(\mathcal{A}, q) \models (agent_2, name_2, g \wedge g_2, tr \wedge tr_2, C_2, R_2) \diamond \ldots \diamond$
$(\mathcal{A}, q) \models (agent_k, name_k, g \wedge g_k, tr \wedge tr_k, C_k, R_k)$

(9) $(\mathcal{A}, q) \models (\epsilon, name, g, tr, (agent_1, name_1, g_1, tr_1, \mathcal{D}($
$(\epsilon, name_{11}, g_{11}, tr_{11}, C_{11}, \epsilon)\, \mathcal{REF}\, (\epsilon, name_{12}, g_{12}, tr_{12}, C_{12}, \epsilon)\, \mathcal{REF} \ldots \mathcal{REF}$
$(\epsilon, name_{1m}, g_{1m}, tr_{1m}, C_{1m}, \epsilon)), R_1)\, Seq\, (agent_2, name_2, g_2, tr_2, C_2, R_2)$
$Seq \ldots Seq\, (agent_k, name_k, g_k, tr_k, C_k, R_k), \epsilon)$ **iff**
$(\mathcal{A}, q) \models (\epsilon, name, g, tr, (\epsilon, name_1, g_1, tr_1, (agent_1, name_{11}, g_{11}, tr_{11},$
$\mathcal{D}(C_{11}), R_1)\, \mathcal{REF}\, (agent_1, name_{12}, g_{12}, tr_{12}, \mathcal{REF}(C_{12}), R_1)$
$\mathcal{REF} \ldots \mathcal{REF}\, (agent_1, name_{1m}, g_{1m}, tr_{1m}, \mathcal{D}(C_{1m}), R_1), \epsilon)\, Seq$
$(agent_2, name_2, g_2, tr_2, C_2, R_2) Seq \ldots Seq\, (agent_k, name_k, g_k, tr_k, C_k, R_k), \epsilon)$

In line (7) we have that $\mathcal{D} \in \{ O, P, F \}$ and $\mathcal{REF} \in \{ And, Or, Seq \}$, 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 $\mathcal{REF}$ of these new subcontracts.

In line (8) we have that $\mathcal{RE} \in \{And, OR\}$ and $\diamond \in \{ \wedge, \vee \}$. 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 ($\wedge$ for AND-refinement) or disjunction ($\vee$ 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 $g \wedge g_k$ and $tr \wedge tr_k$).

Line (9) corresponds to the satisfaction rule for a SEQ-refinement when the first element of the sequence is a deontic norm $\mathcal{D} \in \{ O, P, F \}$ applied over another refinement of subcontracts $\mathcal{REF} \in \{ And, Or, Seq \}$. 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 $\mathcal{RE}\mathcal{F}$ we have before but now combining these new subcontracts.

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

(10) $(\mathcal{A}, q) \models (\epsilon, name, g, tr, (\epsilon, name_1, g_1, tr_1, (agent_{11}, name_{11}, g_{11}, tr_{11}, C_{11}, R_{11})$
$\mathcal{RE}\,(agent_{12}, name_{12}, g_{12}, tr_{12}, C_{12}, R_{12})\,\mathcal{RE}\ldots\mathcal{RE}$
$(agent_{1m}, name_{1m}, g_{1m}, tr_{1m}, C_{1m}, R_{1m}), \epsilon)\,Seq\,(agent_2, name_2, g_2, tr_2, C_2, R_2)$
$Seq\ldots Seq\,(agent_k, name_k, g_k, tr_k, C_k, R_k), \epsilon)$ **iff**
$(\mathcal{A}, q) \models (\epsilon, name, g, tr, ((agent_{11}, name_{11}, g_{11} \wedge g_1, tr_{11} \wedge tr_1, C_{11}, R_{11})\,Seq$
$(agent_2, name_2, g_2, tr_2, C_2, R_2)\,Seq\ldots Seq\,(agent_k, name_k, g_k, tr_k, C_k, R_k))$
$\mathcal{RE}\,((agent_{12}, name_{12}, g_{12} \wedge g_1, tr_{12} \wedge tr_1, C_{12}, R_{12})\,Seq$
$(agent_2, name_2, g_2, tr_2, C_2, R_2)\,Seq\ldots Seq\,(agent_k, name_k, g_k, tr_k, C_k, R_k))$
$\mathcal{RE}\ldots\mathcal{RE}\,((agent_{1m}, name_{1m}, g_{1m} \wedge g_1, tr_{1m} \wedge tr_1, C_{1m}, R_{1m})\,Seq$
$(agent_2, name_2, g_2, tr_2, C_2, R_2)\,Seq\ldots Seq\,(agent_k, name_k, g_k, tr_k, C_k, R_k)), \epsilon)$

(11) $(\mathcal{A}, q) \models (\epsilon, name, g, tr, (\epsilon, name_1, g_1, tr_1, (agent_{11}, name_{11}, g_{11}, tr_{11}, C_{11}, R_{11})$
$Seq\,(agent_{12}, name_{12}, g_{12}, tr_{12}, C_{12}, R_{12})\,Seq\ldots Seq$
$(agent_{1m}, name_{1m}, g_{1m}, tr_{1m}, C_{1m}, R_{1m}), \epsilon)\,Seq\,(agent_2, name_2, g_2, tr_2, C_2, R_2)$
$Seq\ldots Seq\,(agent_k, name_k, g_k, tr_k, C_k, R_k), \epsilon)$ **iff**
$(\mathcal{A}, q) \models (\epsilon, name, g, tr, (agent_{11}, name_{11}, g_{11} \wedge g_1, tr_{11} \wedge tr_1, C_{11}, R_{11})\,Seq$
$(agent_{12}, name_{12}, g_{12} \wedge g_1, tr_{12} \wedge tr_1, C_{12}, R_{12})\,Seq\ldots Seq$
$(agent_{1m}, name_{1m}, g_{1m} \wedge g_1, tr_{1m} \wedge tr_1, C_{1m}, R_{1m})\,Seq$
$(agent_2, name_2, g_2, tr_2, C_2, R_2)\,Seq\ldots Seq\,(agent_k, name_k, g_k, tr_k, C_k, R_k), \epsilon)$

Lines (10) corresponds to the satisfaction rule for a SEQ-refinement when the first element of the sequence is a refinement $\mathcal{RE} \in \{And, OR\}$ 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 $\mathcal{A} = (N, n_0, E, I)$ with associated timed labelled transition system $(Q, q_0, \rightarrow)$ satisfies a *C-O Diagram* C is quite straightforward. It returns "YES" if $(\mathcal{A}, q_0) \models C$, 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:

$(\epsilon, Second\_Update, \epsilon, \epsilon, (Software\,provider, Sends\_Update2, \epsilon, tr_4, O(a_4), R_4)$
$Seq\,(\epsilon, Client\_Second\_Behavior, \epsilon, \epsilon, (Client, Second\_Payment, \epsilon, \epsilon, O(a_5), \epsilon)$
$And\,(Client, Second\_Changes, \epsilon, \epsilon, F(a_6), R_6), \epsilon), \epsilon)$

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}.

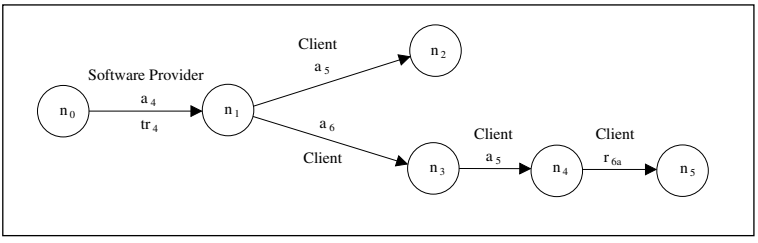

Fig. 3. Automaton $\mathcal{A}$ of *Example 1*

We want to check if the timed automaton $\mathcal{A}$ 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 $n_1 \longrightarrow n_2$ it is fulfilled, but for path $n_1 \longrightarrow n_3$ 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 $\mathcal{A}$.

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

1. Alur, R., Dill, D.L.: Automata For Modeling Real-Time Systems. In: Paterson, M. (ed.) ICALP 1990. LNCS, vol. 443, Springer, Heidelberg (1990)
2. Cambronero, M.E., Valero, V., Díaz, G., Martínez, E.: Web Services Choreographies Verification. Technical Report DIAB-09-04-1, University of Castilla-La Mancha (2009)
3. Lomuscio, A., Qu, H., Solanki, M.: Towards verifying contract regulated service composition. In: Proceedings of IEEE International Conference on Web Services (ICWS 2008), pp. 254–261 (2008)
4. Martínez, E., Díaz, G., Cambronero, M.E., Schneider, G.: A Model for Visual Specification of e-Contracts. In: Proceedings of the 7th IEEE 2010 International Conference on Services Computing (SCC 2010), pp. 1–8 (2010)
5. Prisacariu, C., Schneider, G.: A Formal Language for Electronic Contracts. In: Bonsangue, M.M., Johnsen, E.B. (eds.) FMOODS 2007. LNCS, vol. 4468, pp. 174–189. Springer, Heidelberg (2007)