Table of Contents

On the formal representation of norms and contracts ................................................. 1
Marek Sergot

Contract-based Slicing ..................................................................................................... 3
Daniela da Cruz, Pedro Henriques and Jorge Sousa Pinto

A Calculus of Contracting Processes ............................................................................ 11
Massimo Bartoletti and Roberto Zunino

Contract-based Reasoning about Progress: Application to Resource Sharing in a Network ...... 19
Imene Ben-Hafaiedh, Susanne Graf and Sophie Quinton

Contract-Based Discovery and Adaptation of Web Services ........................................ 27
Luca Padovani

A Survey of Formal Languages for Contracts ............................................................... 29
Tom Hvitved

On the Formal Specification of Business Contracts and Regulatory Compliance ............. 33
Aman Elgammal, Oktay Turetken, Willem-Jan van den Heuvel and Mike Papazoglou

Formal Model for Business-aware Transaction Management ....................................... 37
Francois Hantry, Mohand-Said Hacid and Mike Papazoglou

Autonomous and yet Secure Evolution for Smart Cards Applications using Self-Certification ...... 41
Olga Gadyatskaya and Fabio Massacci

Synthesis and Analysis of Adaptors through Security Contracts ..................................... 49
José Antonio Martin and Ernesto Pimentel

Visual Specification of Formal e-Contracts ..................................................................... 55
Enrique Martinez, Gregorio Diaz and M. Emilia Cambronero

A Trace-based Model for Multi-party Contracts ................................................................ 63
Tom Hvitved

Visual Behavioural Modelling with Contracts ............................................................... 71
Nuno Amálio and Pierre Kelsen

Controllability and Abstraction in Business Processes .................................................. 75
Maria Grazia Buscemi and Hernán Melgratti

Leveraging a Contract-based Synchronization Framework for Design and Analysis of Services ...... 79
Laura K. Dillon, Yi Huang and R.E.K. Stirewalt
Visual Specification of Formal e-Contracts *

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

In this work we present a visual model called C-O Diagrams for the specification of e-contracts. This model includes specification of obligations, permissions, prohibitions, reparations and restrictions such as real time constraints. We also define a formal semantics of the visual model intended for the analysis and verification of the modeled contracts.

1 Introduction

We consider e-contracts as a set of clauses that must be satisfied by several parties participating in a system. These clauses regulate how participants should behave, what are the penalties in case of misbehavior, and under which conditions such clauses must be enacted. We also have that when a clause is violated, if the clause defines a reparation (secondary clauses that come into force when the main clause is not satisfied), and this reparation is fulfilled, then the clause is eventually fulfilled. Recently some works about specifying e-contracts in a formal manner have been released [2, 3, 10], but these approaches consist of formal languages which are hard to study and manipulate by untrained final users of this technology, as business process developers.

The goal of this work is to introduce a new approach for the specification of e-contracts in a user friendly way. E-contracts may be complex, consisting of composite clauses making reference to other clauses in the same or in another contract. Furthermore, we consider contracts with timed restrictions and conditions under which the contract clauses must be applied. Hence, our approach is based on a visual model that we call Contract-Oriented Diagram or C-O Diagram for short, since it is well-known that the use of visual models makes easier the perception of knowledge, and in this way, the intuitive understanding, reading and maintenance of complex problems [6]. This approach can be useful in service-oriented architectures, component-based systems, requirements acquisition, software product lines, etc. In the following sections we first informally define the elements and structure of our visual model, next we formally define the syntax of the diagrams, and finally we present a first version of the formal semantics of these diagrams, based on timed automaton.

2 Visual Model

In our visual model we define a hierarchical tree diagram used to specify the contract clauses that we call C-O Diagram. In Figure 1 we show the basic element. It corresponds to a contract clause and we call it box. This box consists of four fields, allowing us to specify normative aspects or simple norms (P), reparations (R), conditions (g) and time restrictions (tr). Each box has 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

*Partially supported by the Spanish government (cofinanced by FEDER funds) 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.
agent indicates who is the performer of the action (which can be a service, a component or something else, depending on the field where we are applying C-O Diagrams).

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. The time restriction $tr$ specifies the time frame in which the contract clause must be satisfied.

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 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 another contract that must be satisfied in case the main norm is not satisfied, considering the clause eventually satisfied if this reparation is satisfied.

These basic elements of a C-O Diagram can be refined by using AND/OR/SEQ refinements, as shown in Figure 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. A SEQ-refinement means that the norm specified in the target box (SubClause2 in Figure 2) must be fulfilled after satisfying the norm specified in the source box (SubClause1 in Figure 2). In this way, we can build a hierarchical tree with the clauses defined by the 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 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.

In this work, we follow an ought-to-do approach, i.e., the normative aspects of obligations, permissions and prohibitions, as defined in deontic logic [9], are applied over actions performed by the participants in the contract. We only allow the specification of atomic actions in the $P$ field of the leaf clauses of our diagrams. These actions are denoted by lower case Latin letters ("a","b","c",...). We use a dash ("-"") to denote that there is no action specified in the no leaf clauses. The composition of actions can be achieved by means of the different kinds of refinement. In this way, an AND-refinement can be used to model concurrency ";&" between actions, an OR-refinement can be used to model a choice ";+" between actions, and a SEQ-refinement can be used to model sequence ";;" of actions. In Figure 3 we can see an example about how to model these compound actions through refinements, given two atomic actions $a$ and $b$. 
The deontic norms (obligations, permissions and prohibitions) that are applied over these actions can be specified in any clause of our C-O Diagrams, affecting all the actions in the leaf clauses that are subclauses of this clause, but at least one and only one deontic norm must be specified in each one of the branches of our diagram. If it is the case that the clause where we specify the deontic norm is a leaf clause, the norm only affects the atomic action we have in this clause. We use an upper case “O” to denote an obligation, an upper case “P” to denote a permission, and an upper case “F” to denote a prohibition (forbidden). These letters are written in the top left corner of field P. The composition of deontic norms is also achieved by means of the different refinements we have in C-O Diagrams. Thus, an AND-refinement corresponds to the conjunction operator “∧” between norms, an OR-refinement corresponds to the choice operator “+” between norms, and a SEQ-refinement corresponds to the sequence operator “;” between norms. For example, we can imagine having a leaf clause specifying the obligation of performing an action a, written as O(a), and another leaf clause specifying the obligation of performing an action b, written as O(b). These two norms can be combined in the three different ways mentioned before through the different kinds of refinement (Figure 4).

The field R is only allowed in the boxes of our diagrams where we specify a deontic norm of obligation or prohibition in field P, being always empty in the other boxes. This reparation is a new contract that can be just an obligation over an atomic action, but it can also be a more complex diagram, including their own reparations. In this way, we are able to specify nested reparations in our C-O Diagrams. Let us consider a simple contract C stating that we have the obligation of performing an atomic action a and the prohibition of performing an atomic action b. However, if we do not perform the obligatory action a, we can compensate it by fulfilling another contract (modeled by another diagram) called C1, consisting of performing an action c or an action d, and if we perform the forbidden action b, we can compensate it just by performing an action e. This situation can be modeled in our diagrams as shown in Figure 5.

When we specify a guard g and/or a time restriction tr in a clause that is not a leaf clause, they affect all the subclauses, i.e., all the subclauses necessary to satisfy the parent clause must fulfilled the conditions and time restrictions specified in this parent clause. Otherwise, the parent clause is unfulfilled.

---

1 A more detail description of C-O Diagrams can be found in [8], including a case study showing how to specify a concrete e-contract and a evaluation of the model.

---

Figure 3: Composition of actions in C-O Diagrams

Figure 4: Composition of deontic norms in C-O Diagrams
3 Formal Grammar

In this section we formally define the syntax of C-O Diagrams in order to make them susceptible to formal analysis.

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 a finite set of identifiers \( N \) for names. The greek letter \( \epsilon \) means that an expression is empty.

We use \( C \) to denote the contract modeled by a C-O Diagram. The diagrams syntax is defined by the following EBNF grammar:

\[
C := (\text{agent, name, } g, \text{tr}, O(C_2), R) | (\text{agent, name, } g, \text{tr}, P(C_2), \epsilon) | (\text{agent, name, } g, \text{tr}, F(C_2), R) | (\epsilon, \text{name, } tr, C_1, \epsilon)
\]

\[
C_1 := (\text{agent, name, } g, \text{tr}, O(C_2), R) | (\text{agent, name, } g, \text{tr}, F(C_2), R) | (\epsilon, \text{name, } tr, C_1, \epsilon)
\]

\[
C_2 := a | C_3 (\text{And } C_3) | C_3 (\text{Or } C_3) | C_3 (\text{Seq } C_3)
\]

\[
C_3 := (\epsilon, \text{name, } tr, C_1, \epsilon)
\]

\[
R := C | \epsilon
\]

where \( a \in \Sigma \), \( \text{agent} \in {A} \) and \( \text{name} \in N \). 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 \( tr \) 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 most simple contract we can have in C-O Diagrams is that composed of only one box including the elements \( \text{agent} \) and \( \text{name} \). 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 general contract \( C \) as a reparation. For example, \( C := (\text{Buyer, Example1}, \epsilon, \epsilon, O(\text{pay})), C' \) is a very simple contract specifying for a buyer the obligation of paying, otherwise contract \( C' \) comes into effect.

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. For instance, \( C := (e, \text{Example2}, e, x < 5, C' \lor C'', e) \) is a composed contract specifying that contract \( C' \) or contract \( C'' \) must be satisfied in order to satisfy \( C \) before clock \( x \) reach the value 5.

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). Then, these subboxes cannot specify an agent nor a reparation, but it is possible to specify a guard \( g \) and a time restriction \( tr \) affecting only the action in the subbox or the action composed in its refinements. For example, \( C := (\text{Buyer, Example3}, e, e, O(C' \lor C''), e) \), where we have that \( C' := (e, \text{Option1}, e, e, \text{pay cash}, e) \) and \( C'' := (e, \text{Option2}, e, e, \text{pay card}, e) \), is a contract specifying for a buyer the obligation of paying by cash or by credit card.

4 Formal Semantics

The \( C-O \) Diagrams formal semantics is given by means of a satisfaction function. To define this satisfaction function 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 [4] for a complete definition of timed automaton and its semantics.

Definition 2 (C-O Diagrams Semantics)

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 \( (\mathcal{A}, q) \models C \) (\( C \) in state \( q \) satisfies contract \( C \)) as follows:

1. \((\mathcal{A}, q) \models (\text{agent}, \text{name}, g, tr, O(a), R)\) iff \( \forall \langle q_1 \rightarrow q_2 \rightarrow \ldots \rightarrow q_j \rangle \) for \( q = q_1 \):

   - The main clause holds, that is, \( \exists \bar{i} \in [1, j - 1] \) such that \( q_i \xrightarrow{a} q_{i+1} \) with \( n_i \xrightarrow{g,a,r} n_{i+1} \) where \( (g \land tr) \in gt \) and agent\((a)\)

2. \((\mathcal{A}, q) \not\models (\text{agent}, \text{name}, g, tr, P(a), e)\) iff \( \exists \langle q_1 \rightarrow q_2 \rightarrow \ldots \rightarrow q_j \rangle \) for \( q = q_1 \) where the main clause holds, that is, \( \exists \bar{i} \in [1, j - 1] \) such that \( q_i \xrightarrow{a} q_{i+1} \) with \( n_i \xrightarrow{g,a,r} n_{i+1} \) where \( (g \land tr) \in gt \) and agent\((a)\)
(3) $(\mathcal{A}, q) \models (agent, name, g, tr, F(a), R)$ iff $\forall (q_1 \rightarrow q_2 \rightarrow \ldots \rightarrow q_j)$ 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 $n_i \xrightarrow{g, a, r} n_{i+1}$ where $(g \land tr) \in g^t$ and agent($a$),

The main clause does not hold but reparation holds, that is, $R \not= \varepsilon$ and $(\mathcal{A}, q_{i+1}) \models R$ for the first $i \in [1, j-1]$ such that $q_i \xrightarrow{a} q_{i+1}$ with $n_i \xrightarrow{g, a, r} n_{i+1}$ where $(g \land tr) \in g^t$ and agent($a$)

(4) $(\mathcal{A}, q) \models (agent, name, g, tr, D((\varepsilon, name_1, g_1, tr_1, C_1, \varepsilon) \land \varepsilon, \ldots, (\varepsilon, name_k, g_k, tr_k, C_k, \varepsilon), R))$ iff

$(\mathcal{A}, q) \models (\varepsilon, name, g, tr, (agent_1, name_1, g_1, tr_1, C_1, R_1) \land \ldots \land (agent_k, name_k, g_k, tr_k, C_k, R_k), \varepsilon)$

(5) $(\mathcal{A}, q) \models (agent_1, name_1, g \land g_1, tr \land tr_1, C_1, R_1) \land \ldots \land (agent_k, name_k, g \land g_k, tr \land tr_k, C_k, R_k)$

□

Lines (1)–(3) correspond to the semantics 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 $(g \land tr) \in g^t$). 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 $(u + d) \not\in tr$). 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.

In lines (4)–(5) we have that $\mathcal{D} \in \{O, P, F\}$, $\mathcal{R} \in \{And, Or\}$ and $\varnothing \in \{\land, \lor\}$. Line (4) corresponds to the semantics of applying a deontic norm over an AND-refinement or an OR-refinement. 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 $\mathcal{R} \in$ of these new subcontracts. Line (5) corresponds to the semantics of these refinements when no deontic operator is applied over them (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 $tr \land tr_k$).

In this C-O Diagrams semantics we deliberately omit SEQ-refinement. The semantics of this refinement is more complex because we have to guarantee that the subcontracts not only are fulfilled but they are also fulfilled in the correct order. For that purpose we are now working on new semantic rules to apply distributivity of SEQ-refinement over the other refinements, so we can finally have sequence of deontic norms applied over atomic actions. In this way, we can determine if in the location we reach after satisfying the first element of the sequence the remaining sequence is fulfilled.
5 Conclusions and Future Work

In this paper we have presented C-O Diagrams, a new visual formalism for electronic contracts. To the best of our knowledge, there is not any other visual model specially created for the definition of e-contracts. There are several works in the literature that define a meta-model for the specification of e-contracts based on some already existing visual specification like UML or entity-relationship diagrams [5, 7], but they all lack of some of the capabilities we have in C-O Diagrams (real time constraints, reparations, . . .).

As future work, apart from complete C-O Diagrams semantics, we are planning to apply this model to different case studies in order to evaluate its usefulness in different fields. We also envisage the possibility of specifying a different C-O Diagram for each one of the parties involved in an e-contract instead of having a global C-O Diagram with multiple agents. This compositional approach can be useful if we define a composition operator, specifying when two of these new C-O Diagrams can be composed and the result of the composition.

References


