Contracts for Interacting Two-Party Systems

Gordon J. Pace
Department of Computer Science,
University of Malta
gordon.pace@um.edu.mt

Fernando Schapachnik*
Departamento de Computación, FCEyN,
Universidad de Buenos Aires, Buenos Aires, Argentina
fschapachnik@dc.uba.ar

This article deals with the interrelation of deontic operators in contracts – an aspect often neglected when considering only one of the involved parties. On top of an automata-based semantics we formalise the onuses that obligations, permissions and prohibitions on one party impose on the other. Such formalisation allows for a clean notion of contract strictness and a derived notion of contract conflict that is enriched with issues arising from party interdependence.

1 Introduction

Deontic modalities such as permission and obligation have been debated exhaustively in the literature, and various formalisms exist with different interpretations and axiomatisation of deontic notions. With few exceptions, the modalities are usually presented in an impersonal manner, refering only to the subject of the modality. For instance, most formalisms enable reasoning about notions such as "John is permitted to withdraw cash" and "John is obliged to pay an annual credit card fee". However, in a contractual setting, the behaviour involves interaction between the two parties the contract binds, and such statements about the ideal behaviour have both a notion of the subject and object of the action. For instance, in a contract between John and his bank, the clause "John is permitted to withdraw cash" is about both parties, and can be interpreted to mean that if John attempts to withdraw cash, then the bank will not refuse or hinder his action. Similarly, the clause "John is obliged to pay an annual credit card fee" places an obligation on John to perform an action with the bank as the object of the action, and (arguably) also places the onus on the bank to accept the payment. Interacting parties allow for both cooperation and interference between the parties in the actions they perform, and thus bring about an additional dimension to contract analysis. An interesting corollary to this view, is that permission can now be seen as a first class deontic modality. Typically seen as the dual of prohibition, violations of permissions have always proved difficult to formalise their violation, mainly since a branching logic analysis is required (if party p were to perform a then they would not be stopped from doing so). In an interacting two party system context, permission now takes a first class role, obliging the object of the modality to allow the subject to perform the action if they so desire.

Although the work on deontic logic for interacting parties is not abundant, computer scientists have studied for various decades concurrent and synchronous composition, notions which embody precisely interaction from an action-based perspective. In [11] we have presented work-in-progress on how synchrony can be applied in a contractual setting, using a formal automaton-based model of interacting two-party systems in which the parties synchronise over a set of actions. In this paper we extend the work presented there to deal with (i) absence of actions; (ii) mutually exclusive actions; (iii) conflicts.

The rest of the paper is organised as follows. The next section formalises our notions of automata, deontic operators, contracts and contracts' strength, which allows us to show, in Section 3 that some

*Partially founded by UBACyT 20020100200103.

To appear in EPTCS.

contracts cannot be satisfied at the same time and thus lead to a conflict. Finally, in Section 4 we discuss related work, and conclude in Section 5.

2 Regulated Two-Party Systems

2.1 An Automata-Based View

To enable direct reasoning about contracts, one requires a model in which the two parties somehow interact to agree on which actions to perform. We use the notion of synchronous composition [1] to model such behaviour. Furthermore, to be able to deal with concurrent obligations (for instance, one party being obliged to perform one action and the other being obliged to perform another), we adopt multi-action labels on transitions, since if we do not, it would be impossible not to violate a contract in which both parties have different obligations at the same time.

Definition 1 *A multi-action automaton S is a tuple $\langle \Sigma, Q, q0, \rightarrow \rangle$, where Σ is the alphabet of actions, Q is the set of states, $q0 \in Q$ is the initial state and $\rightarrow \subseteq Q \times 2^{\Sigma} \times Q$ is the transition relation. We will write $q \xrightarrow{A} q'$ for $(q,A,q') \in \rightarrow$, next(q) to be the set of target state and action set pairs of transitions outgoing from q (defined to be $\{(A,q') \mid q \xrightarrow{A} q'\}$) and acts$(q)$ to be the set of all action sets on the outgoing transitions from q (defined to be $\{A \mid \exists q' \cdot q \xrightarrow{A} q'\}$). We say that an automaton is total, if for every $q \in Q$ and $A \subseteq \Sigma$, there is a $q' \in Q$ such that $q \xrightarrow{A} q'$.*

The synchronous composition of two automata $S_i = \langle Q_i, q0_i, \rightarrow_i \rangle$ for $i \in \{1,2\}$ (both with alphabet Σ) synchronising over alphabet G, written $S_1 \|_G S_2$, and is defined to be $\langle Q_1 \times Q_2, (q0_1, q0_2), \rightarrow \rangle$, where $\rightarrow$ is the classical synchronous composition relation defined below:

$$\frac{q_1 \xrightarrow{A}_1 q_1'}{(q_1,q_2) \xrightarrow{A} (q_1',q_2)}\, A \cap G = \emptyset \qquad \frac{q_2 \xrightarrow{A}_2 q_2'}{(q_1,q_2) \xrightarrow{A} (q_1,q_2')}\, A \cap G = \emptyset$$

$$\frac{q_1 \xrightarrow{A}_1 q_1',\ q_2 \xrightarrow{B}_2 q_2'}{(q_1,q_2) \xrightarrow{A \cup B} (q_1',q_2')}\, A \cap G = B \cap G \neq \emptyset$$

We can now define contracts to be automata with each state tagged with the contract which will be in force at that point. The contracts will be able to refer to both presence and absence of an action. Given an alphabet of actions Σ, we write $\Sigma!$ to refer to the alphabet extended with actions preceded with an exclamation mark ! to denote their absence: $\Sigma! \stackrel{df}{=} \Sigma \cup \{!a \mid a \in \Sigma\}$. We use variables x and y to range over $\Sigma!$. If x is already an inverted action $x = !a$, then expression $!x$ is interpreted to be a.

Contract clauses are either (i) obligation clauses of the form $\mathscr{O}_p(a)$ or $\mathscr{O}_p(!a)$, which say that party p is obliged to perform or not perform action a respectively; or (ii) permission clauses which can be either of the form of $\mathscr{P}_p(a)$ or $\mathscr{P}_p(!a)$ (party p is permitted to perform, or not perform action a respectively).

Definition 2 *A* contract clause *over alphabet Σ is structured as follows (where action $x \in \Sigma!$, party $p \in \{1,2\}$):*

$$\mathit{Clause} ::= \mathscr{O}_p(x) \mid \mathscr{P}_p(x)$$

A contract automaton *is a total and deterministic multi-action automaton $S = \langle Q, q0, \rightarrow \rangle$, together with a total function contract $\in Q \rightarrow 2^{\mathit{Clause}}$ assigning a set of clauses to each state. We use $\mathscr{CA}$ to refer to the class of contract automata.*

Two contract automata are said to be structurally isomorphic if they are structurally identical automata (they have the same set of states, initial state and transition relation) but may have different contract functions.

Structurally isomorphic contract automata allow us to reason about the weakening or strengthening of a contract by changing the clauses in particular states but respecting the structure (and thus the temporal behaviour) of the contract, and will be used in various theorems in the rest of the paper. We can now define a regulated two-party system in terms of multi-action automata.

Definition 3 *A* regulated two-party system *synchronising over the set of actions G is a tuple $R = \langle S_1, S_2 \rangle^{\mathscr{A}}_{G}$, where $S_i = (\Sigma_i, Q_i, q0_i, \rightarrow_i)$ is a multi-action automaton specifying the behaviour of party i, and $\mathscr{A}$ is a contract automaton over alphabet $\Sigma_1 \cup \Sigma_2$.*

The behaviour of a regulated two-party system R, written $[\![R]\!]$, is defined to be the automaton $(S_1 \|_G S_2) \|_\Sigma \mathscr{A}$. To make states in such systems more readable, we will write $((q_1, q_2), q_{\mathscr{A}})$ as $(q_1, q_2)_{q_{\mathscr{A}}}$.

A regulated two-party system is well-formed if $S_1 \|_G S_2$ never deadlocks: $\forall (q_1, q_2) \cdot acts(q_1, q_2) \neq \emptyset$.

In the rest of the paper we will assume that all systems are well-formed, i.e., do not deadlock. One way of guaranteeing this may be by having all system states provide a transition with the empty action.
Also note that the totality of the contract automaton guarantees that the system behaviour is not constrained, but simply acts to tag the states with the relevant contracts at each point in time.

2.2 Contract Satisfaction

Given a two-party system (S_1, S_2), and a contract automaton $\mathscr{A}$, we can now define whether or not either party is violating the contract when a particular state is reached or a transition is taken. As we will see, a dual-view of violation, identifying *both* bad states and bad transitions, is necessary in a deontic context. We will look at the different deontic operators and define the set of violations induced for each of them.

Definition 4 *Functions $O_p(q_{\mathscr{A}})$ and $F_p(q_{\mathscr{A}})$ give the set of actions respectively obliged to be performed and obliged not to be performed by party p. They are defined in terms of the contract clauses in the state.*

$$O_p(q_{\mathscr{A}}) \stackrel{df}{=} \{a \mid \mathscr{O}_p(a) \in contract(q_{\mathscr{A}})\}$$
$$F_p(q_{\mathscr{A}}) \stackrel{df}{=} \{a \mid \mathscr{O}_p(!a) \in contract(q_{\mathscr{A}})\}$$

Action set A is said to be viable for party p in a contract automaton state $q_{\mathscr{A}}$, written $viable_p(q_{\mathscr{A}}, A)$, if (i) all her obliged actions are included in A but; (ii) no actions which the party is obliged not to perform are included A:

$$viable_p(q_{\mathscr{A}}, A) \stackrel{df}{=} O_p(q_{\mathscr{A}}) \subseteq A \wedge F_p(q_{\mathscr{A}}) \cap A = \emptyset$$

Since we would like to be able to place blame in the case of a violation, we parametrise contract satisfaction and violation by party.

It is also worth noting that while obligation to perform an action, for instance, is violated in a transition which does not include the action, permission is violated by a state in which the opportunity to perform the permitted action is not present. The satisfaction predicate will thus be overloaded to be applicable to both states and transitions. The predicate $sat^{\mathscr{A}}_p(X)$ will denote that the contract automaton $\mathscr{A}$, reaching state X or traversing transition X, does not constitute a violation for party p. X ranges over states and transitions in the composed system. When $\mathscr{A}$ is clear from the context, we simply write $sat_p(X)$. We start by defining separate satisfaction predicates for the deontic operators.

Permission. If party p is permitted to perform shared action a, then the other party $\overline{p}$ must provide p with at least one viable outgoing transition which contains a but does not include any forbidden actions (that is, it is *viable* for p). Permission to perform local actions cannot be violated. In the case of a single permission, this can be expressed as follows:

$$(q_1,q_2)_{q_{\mathscr{A}}} \vdash_p \mathscr{P}_p(a) \stackrel{df}{=} true$$
$$(q_1,q_2)_{q_{\mathscr{A}}} \vdash_{\overline{p}} \mathscr{P}_p(a) \stackrel{df}{=} a \in G \implies \exists A \in acts(q_{\overline{p}}),\, A' \subseteq G^c \cdot a \in A \wedge viable_p(q_{\mathscr{A}}, A \cup A')$$

Similarly, if party p is permitted to not perform action a, then the other party $\overline{p}$ must provide p with at least one viable outgoing transition which does not include a nor any forbidden action. Permission to perform local actions can never be violated. In the case of a single permission, this can be expressed as follows:

$$(q_1,q_2)_{q_{\mathscr{A}}} \vdash_p \mathscr{P}_p(!a) \stackrel{df}{=} true$$
$$(q_1,q_2)_{q_{\mathscr{A}}} \vdash_{\overline{p}} \mathscr{P}_p(!a) \stackrel{df}{=} a \in G \implies \exists A \in acts(q_{\overline{p}}),\, A' \subseteq G^c \cdot a \notin A \wedge viable_p(q_{\mathscr{A}}, A \cup A')$$

While actual obligation violations occur when an action is not performed, violations of a permission occur when no appropriate action is possible. For any other parameters, the permission is otherwise satisfied.

Example: If p is permitted to withdraw money from the bank, permitted not to deposit, obliged to pay the fee, and obliged not to steal ($\mathscr{P}_p(w)$, $\mathscr{P}_p(!d)$, $\mathscr{O}_p(f)$, $\mathscr{O}_p(!s)$), $\overline{p}$ should provide at least one transition that contains both a w and an f and contains neither a d nor an s.

To combine all permissions in a state, we simply take the conjunction of all conditions:

$$sat^P_p((q_1,q_2)_{q_{\mathscr{A}}}) \stackrel{df}{=} \forall \mathscr{P}_{\overline{p}}(x) \in q_{\mathscr{A}} \cdot (q_1,q_2)_{q_{\mathscr{A}}} \vdash_p \mathscr{P}_{\overline{p}}(x)$$

All transitions are taken as satisfying the permission satisfaction function.

Obligation. Obligation brings in constraints on both parties. Given that party p is obliged to perform action a in a state means that (i) party p must include the action in any outgoing transition in the composed system in which it participates; and (ii) the other party $\overline{p}$ must provide a viable synchronisation action set which, together with other asynchronous actions performed by p, allows p to perform *all* its obligations, positive and negative. Obligation to not perform action a ($\mathscr{O}_p(!a)$) can be similarly expressed. We combine all positive and negative obligations in the following definition:

$$sat^O_p((q_1,q_2)_{q_{\mathscr{A}}} \xrightarrow{A} (q_1',q_2')_{q'_{\mathscr{A}}}) \stackrel{df}{=} viable_p(q_{\mathscr{A}}, A)$$
$$sat^O_{\overline{p}}((q_1,q_2)_{q_{\mathscr{A}}}) \stackrel{df}{=} \exists A \in acts(q_{\overline{p}}),\, A' \subseteq G^c \cdot viable_p(q_{\mathscr{A}}, A \cup A')$$

The satisfaction constraint for transitions is only applicable if A is not an action set performed asynchronously by $\overline{p}$. For other parameters, $sat^O_p(X)$ is true.

Example: Continuing the previous example, to satisfy sat^O_p, *all* of p's outgoing transitions must be s-free and must have an f, while $\overline{p}$ should offer at least one transition that contains an f and not an s. That is, if at a given state $\overline{p}$ offers only outgoing transitions labeled $\{f,s\}$ then she is forcing p to an s in order to have an f, and thus not satisfying its part in p's obligations.

General contract satisfaction. It is defined as: $sat_p(X) \stackrel{df}{=} sat_p^P(X) \wedge sat_p^O(X)$. Based on this, we can now define correctness of a regulated two-party system.

Definition 5 *A party p is said to be incapable of breaching a contract in a regulated two-party system $R = \langle S_1, S_2 \rangle_G^{\mathscr{A}}$, written $breachIncapable_p(R)$, if p cannot be in violation in any of the reachable states and transitions of R.*

Note that a party being breach-incapable is stronger than just being compliant for one specific run — $breachIncapable_p(R)$ means that there is no possible trace of R, in which p breaches the contract.

2.3 Other Modalities

Definition 6 *Permissions and obligations are duals under a notion of norm opposites and action absence. We define the opposite of permission and obligation $!\mathscr{P}_p(x)$ and $!\mathscr{O}_p(x)$ syntactically in the following manner:*

- *Party p not being permitted to perform an action is equivalent to p being obliged not to perform the action:* $!\mathscr{P}_p(a) \stackrel{df}{=} \mathscr{O}_p(!a) \qquad !\mathscr{P}_p(!a) \stackrel{df}{=} \mathscr{O}_p(a)$
- *Party p not being obliged to perform an action is equivalent to p being permitted not to perform the action:* $!\mathscr{O}_p(a) \stackrel{df}{=} \mathscr{P}_p(!a) \qquad !\mathscr{O}_p(!a) \stackrel{df}{=} \mathscr{P}_p(a)$

It should be noted that we are equating lack of permission to do a to an obligation to perform an action set which does not include a. Although this seems to go against the intuitive idea of letting a party do nothing as a way of not violating lack of permission, note that (i) since transitions carry sets of actions, the empty set of actions is a way of satisfying the obligation; and (ii) well-formedness (see Definition 3) of the parties ensures that progress is always possible thus making the formulation of lack of permission conform to our expectations.

It is interesting to note that in a two party system there are alternative notions of opposites to permission and obligation. Consider party p not being permitted to perform action a. Apart from the interpretation we gave, in which the norm places the onus on party p not to perform a, an alternative view is to push the responsibility to $\overline{p}$ and interpret it as: *party $\overline{p}$ may not provide a viable action set which includes a*. This is distinct from $!\mathscr{P}_p(a)$ (and indeed from the other modalities we have). Similarly, consider party p not being obliged to perform action a. The interpretation we adopted permits party p to not perform a, but once again, alternative definitions may be adopted. One possibility is to push the responsibility to $\overline{p}$ and interpret it as: *party $\overline{p}$ must provide a viable transition which does not include a*. These duals, in which the outer negation of a norm also corresponds to shifting of responsibility give an interesting alternative view of norm opposites in a two-party system. Another interesting alternative would be to interpret these negations as modalities whose only effect is the cancelling of existing clauses. We will not explore these alternative modalities any further in this paper, since the modalities we adopt provide a clean notion of conflicts, as discussed in Section 3. Should they be needed for a particular application, any of the above mentioned interpretations could be included as alternative type of negation. One of the advantages of clear formal semantics is that there is no need to dispute the meaning of a given term, since different ones can be defined and the appropriate one be picked to convey specific meanings. Prohibition can now be defined as the dual of permission:

Definition 7 *Prohibition contract clauses $\mathscr{F}_p(a)$ and $\mathscr{F}_p(!a)$, prohibiting party p from performing and not performing a respectively, can be expressed in terms of permission:*

$$\mathscr{F}_p(a) \stackrel{df}{=} !\mathscr{P}_p(a) \qquad \mathscr{F}_p(!a) \stackrel{df}{=} !\mathscr{P}_p(!a)$$

These definitions allow us to express prohibition in terms of obligation not to perform an action:

Proposition 1 *Prohibition to perform an action is equivalent to obligation not to perform the action:* $\mathscr{F}_p(x) = \mathscr{O}_p(!x)$.

2.4 Contract Strength

We can now define strictness relationships over contracts.

Definition 8 *A contract automaton* $\mathscr{A}'$ *is said to be* stricter than *contract automaton* $\mathscr{A}$ *for party p (or* $\mathscr{A}$ *said to be* more lenient *than* $\mathscr{A}'$ *for party p), written* $\mathscr{A} \sqsubseteq_p \mathscr{A}'$*, if for any systems* S_1 *and* S_2*,* $breachIncapable_p(\langle S_1, S_2 \rangle_G^{\mathscr{A}'}) \implies breachIncapable_p(\langle S_1, S_2 \rangle_G^{\mathscr{A}})$*. We say that two contract automata* $\mathscr{A}$ *and* $\mathscr{A}'$ *are equivalent for party p, written* $\mathscr{A} =_p \mathscr{A}'$*, if* $\mathscr{A} \sqsubseteq_p \mathscr{A}'$ *and* $\mathscr{A}' \sqsubseteq_p \mathscr{A}$*. We define global contract strictness* $\mathscr{A} \sqsubseteq \mathscr{A}'$ *to hold if* $\mathscr{A} \sqsubseteq_p \mathscr{A}'$ *holds for all parties p, and similarly global contract equivalence* $\mathscr{A} = \mathscr{A}'$.

Proposition 2 *The relation over contracts* $\sqsubseteq$ *is a partial order.*

Structurally isomorphic contract automata provide a useful proof technique:

Proposition 3 *Given two structurally isomorphic contract automata* $\mathscr{A}$ *and* $\mathscr{A}'$*,* $\mathscr{A} \sqsubseteq \mathscr{A}'$ *if and only if, for any state or transition X,* $sat_p^{\mathscr{A}'}(X) \implies sat_p^{\mathscr{A}}(X)$.

The full proof of the proposition can be found in [10].

Proposition 4 *Contract automata are monotonic: given two structurally isomorphic contract automata* $\mathscr{A}$ *and* $\mathscr{A}'$*, with contract clause functions contract and contract*$'$ *respectively, which satisfy that* $\forall q \cdot contract(q) \subseteq contract'(q)$*, it follows that* $\mathscr{A} \sqsubseteq \mathscr{A}'$.

Although contracts are expressed as automata, we would like to be able to compare individual clauses. To do this we will need to relate contract automata which are equivalent except for a particular clause replaced by another.

Definition 9 *Given two contract clauses C and* C'*, the relation over contract automata* $[C \to C'] \subseteq \mathscr{CA} \times \mathscr{CA}$ *relates two contract automata* $\mathscr{A}$ *and* $\mathscr{A}'$ *if* $\mathscr{A}$ *is equivalent to* $\mathscr{A}'$ *except possibly for a number of instances of clause C replaced by* C'.

We extend the notion of strictness to contract clauses. We say that clause C' *is stricter than clause C for party p, written* $C \sqsubseteq_p C'$*, if for any contract automata* $\mathscr{A}$ *and* $\mathscr{A}'$ *such that* $(\mathscr{A}, \mathscr{A}') \in [C \to C']$*, it follows that* $\mathscr{A} \sqsubseteq_p \mathscr{A}'$*. We similarly extend the notion of strictness for all parties* $\sqsubseteq$.

The following proposition allows us to use the proof principle given in Proposition 3 for reasoning about clause strictness:

Proposition 5 *Given clauses C and* C'*, any two contract automata related by* $[C \to C']$ *are structurally isomorphic.*

2.5 Strictness Theorems

The strictness relationship between clauses allows us to state the following theorems.

Theorem 1 *Obligation is stricter than permission: (i) $\mathscr{P}_p(a) \sqsubseteq \mathscr{O}_p(a)$; and (ii) $\mathscr{P}_p(!a) \sqsubseteq \mathscr{O}_p(!a)$.*
Proof: *We present the proof of (i) — the proof of (ii) is very similar. We need to prove that for any contract automata $\mathscr{A}$ and $\mathscr{A}'$ such that $(\mathscr{A}, \mathscr{A}') \in [\mathscr{P}_p(a) \to \mathscr{O}_p(a)]$, then it follows that $\mathscr{A} \sqsubseteq \mathscr{A}'$. Using Proposition 5, we know that $\mathscr{A}$ and $\mathscr{A}'$ are structurally isomorphic, allowing us to apply the proof principle of Proposition 3.*

We thus have to show that $sat_p^{\mathscr{A}'}(X)$ implies $sat_p^{\mathscr{A}}(X)$. Since the permission in $\mathscr{A}$ which is replaced by an obligation, never yields violations for party p nor for any party on transitions, it suffices to prove that this implication holds on states for party $\overline{p}$.
The satisfaction function for $\overline{p}$'s obligations in states is:

$$\exists A \in acts(q_{\overline{p}}),\ A' \subseteq G^c \ \cdot\ viable_p(q_{\mathscr{A}'}, A \cup A')$$

If $a \in G$, and since $a \in O_p(q_{\mathscr{A}'})$, we can conclude that $a \in A$:

$$a \in G \implies \exists A \in acts(q_p),\ A' \subseteq G^c \ \cdot\ a \in A \wedge viable_{\overline{p}}(q_{\mathscr{A}'}, A \cup A')$$

Furthermore, since $q_{\mathscr{A}}$ has less obligations than $q_{\mathscr{A}'}$, viability for $q_{\mathscr{A}'}$ implies viability for $q_{\mathscr{A}}$:

$$a \in G \implies \exists A \in acts(q_p),\ A' \subseteq G^c \ \cdot\ a \in A \wedge viable_{\overline{p}}(q_{\mathscr{A}}, A \cup A')$$

Hence, the satisfaction function for the permission $\mathscr{P}_p(a)$ holds and thus, by Proposition 3 we can conclude that $\mathscr{A} \sqsubseteq \mathscr{A}'$.

Theorem 2 *For synchronised actions, obligation for one party is stricter than permission for the other: (i) $\mathscr{P}_p(a) \sqsubseteq \mathscr{O}_{\overline{p}}(a)$; and (ii) $\mathscr{P}_p(!a) \sqsubseteq \mathscr{O}_{\overline{p}}(!a)$.*

It is interesting to note that if we had a weaker semantics which simply identifies a violation without identifying the guilty party, we would be able to show equivalence between $\mathscr{O}_p(a)$ and $\mathscr{O}_{\overline{p}}(a)$, since a lack of a on a transition would cause a violation of both obligations. However, since our semantics characterise violations for the parties separately, and the partial order $\sqsubseteq_p$ is parametrised by the party, we can show that the two obligations are in fact different [11].

2.6 Mutually Exclusive Actions

Although we adopt a multi-action approach, modelling real-world scenarios means that certain actions should never occur concurrently. For instance, one would expect that the automata never perform the action *openDoor* and *closeDoor* on the same transition. This allows us to identify strictness laws which hold only for mutually exclusive actions.

Definition 10 *Given a multi-action automaton $\langle \Sigma, Q, q0, \to \rangle$, two actions a and b ($\{a,b\} \subseteq \Sigma$) are said to be mutually exclusive, written $a \bowtie b$, if they can never appear in the same set of actions on transitions. Thus, for any automaton, it should be the case that:*

$$\forall (q, A, q') \in \to \ \cdot\ a \in A \implies b \notin A$$

In the rest of the article we will assume that mutually exclusive actions never appear in the synchronisation sets. Removing this restriction, however, does not affect the results we present. The following theorem shows how mutually exclusive actions and action absence are related together under both obligation and permission:

Theorem 3 *If $a \bowtie b$ then (i) $\mathscr{O}_p(!a) \sqsubseteq \mathscr{O}_p(b)$; and (ii) $\mathscr{P}_p(!a) \sqsubseteq \mathscr{P}_p(b)$.*

A similar result can be shown, but referring to the other party in the contract:

Theorem 4 *If $a \bowtie b$ then $\mathscr{O}_{\overline{p}}(!b) \sqsubseteq \mathscr{O}_p(a)$.*

Although one may be tempted to induce that a similar result can be shown for permission (analogous to part (ii) of Theorem 3) — $\mathscr{P}_{\overline{p}}(!b) \sqsubseteq \mathscr{P}_p(a)$ does not always hold. As a simple example of a system satisfying $\mathscr{P}_p(a)$ but not $\mathscr{P}_{\overline{p}}(!b)$, consider party p be able to perform just one transition with action set $\{b\}$, and party $\overline{p}$ being able to perform one of two transitions: one with action set $\{a\}$, the other with action set $\{b\}$. Party p is permitted to perform a but party $\overline{p}$ is not permitted to perform $!b$.

3 Conflicts

Contract clauses are not always compatible with one another. Many definitions of conflict are possible — in this article we deal only with one particular class of conflicts which focusses on conflicting norms and mutually exclusive actions, but some interesting issues arise from party interdependence. As expected, the obligation on a party to perform an action a and the obligation on the same party not to perform the same action can never be satisfied together. Another interesting example is that of $\mathscr{P}_p(!a)$ and $\mathscr{O}_p(a)$. Although one is tempted to intuitively think that having the possibility of doing something other than a does not conflict with the obligation of doing a, multi-action semantics in contracts are different: to satisfy the permission party $\overline{p}$ must provide a-free action sets which allow p to satisfy her obligations, but that requires that they contain a. In this section we axiomatise the notion of conflicts in interacting two-party systems and investigate some consequences.

Definition 11 *Contract conflicts is a relation between contract clauses $\maltese \in Clause \leftrightarrow Clause$ and is defined to be the least relation satisfying the following axioms:*
Axiom 1: *Opposite permissions conflict: $\vdash \mathscr{P}_p(x) \maltese !\mathscr{P}_p(x)$.*
Axiom 2: *Obligation to perform mutually exclusive actions is a conflict: $a \bowtie b \vdash \mathscr{O}_p(a) \maltese \mathscr{O}_p(b)$.*
Axiom 3: *Conflicts are closed under symmetry: $C \maltese C' \vdash C' \maltese C$.*
Axiom 4: *Conflicts are closed under increased strictness: $C \maltese C' \wedge C' \sqsubseteq C'' \vdash C \maltese C''$.*

Although conflicts are only identified for opposing permissions in the axioms, they also arise in opposing obligations, and can be shown to follow from the axioms.

Proposition 6 *Opposite obligations conflict with each other: $\mathscr{O}_p(x) \maltese !\mathscr{O}_p(x)$.*

Proposition 7 *Obligation to perform an action conflicts with both permission and obligation to not perform it: (i) $\mathscr{O}_p(x) \maltese \mathscr{P}_p(!x)$; and (ii) $\mathscr{O}_p(x) \maltese \mathscr{O}_p(!x)$. Obligation to perform an action also conflicts with lack of permission to perform the action: (iii) $\mathscr{O}_p(x) \maltese !\mathscr{P}_p(x)$.*

Proposition 8 *Given two conflicting clauses $C_1 \maltese C_2$, making the two clauses stricter does not resolve the conflict: if $C_1 \sqsubseteq C_1'$ and $C_2 \sqsubseteq C_2'$, then $C_1' \maltese C_2'$.*

Example: As a simple example, consider John signing a contract with his bank. The contract says that (i) whenever he is logged into his Internet banking account, he is to be permitted to make money transfers; and (ii) if a malicious attempt to log in to his account is identified, logging in and making transfers will be prohibited until the situation is cleared. The two statements can be expressed in the two contract automata shown in Fig. 1. Combining the two statements, however results in an automaton where initially, after performing action set {login, malicious}, one ends up in a state with both $\mathscr{P}_p$(transfer) and $\mathscr{F}_p$(transfer), which are in conflict.

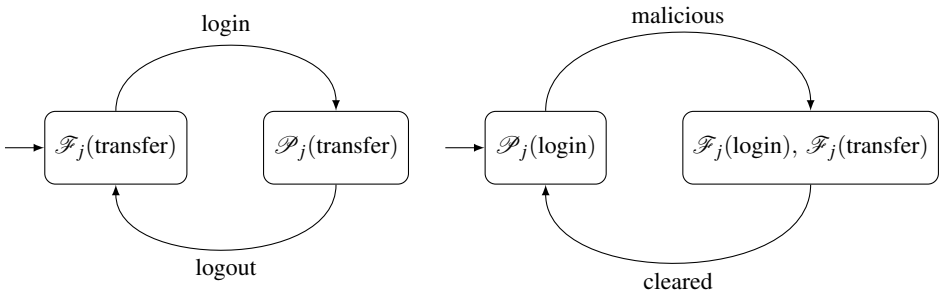

Figure 1: Internet banking contracts

4 Related Work

Despite the fact that contracts are, by definition, an agreement between two or more parties, most formal studies regulate the parties independently and do not analyse how permissions, obligations or prohibitions for one party affect the other, or do so in limited ways. Here we summarise the most related work.

[3] deals with obligation violations in contracts using the domain specific BCL language [4], introducing contrary-to-duty clauses and directed obligations, but does not analyse the reciprocity of deontic clauses in a contract. [9] aims at formalisations of contracts for e-commerce but focuses only on analysing temporal consistency. A related line of research was started by [5], later followed upon by various others ([14, 2], etc.) — although not explicitly about contracts, they look at a flavour of axiomatic deontic logic with obligations being directed from one individual towards another, termed *directed obligations*. Directed permissions have also been studied, but were termed to be conflicting because of lack of a clear counterparty, following both the *claimant theory* or the *benefit theory*. Once one considers actions that are only realisable by the two parties in synchrony, as our approach does, the concept of permission appears more clearly. Although it does not fully consider many aspects of permission e.g., $\mathscr{P}_p(!a)$ – it would be interesting to direct further research to look at the similarities between both approaches, including variations such as [12].

Our model does not provide explicitly for the notion of *interference* that has been analysed by many, notably Hohfeld [6] and Lindahl [8], It is important to understand, however, that the difference between *vested* and *naked* liberties (i.e., warranty of immunity from interference) relates to a real concern in the context of general law but blurs in the context of a contract where one party allowing the other to perform a shared action, but reserving itself the right to interfere, does not have practical sense. More specifically, in our formal model $\mathscr{P}_p(a)$ means not only that p may attempt to perform a — it means that p would succeed in doing a should she try. If the notion of *attempting* to do an action a that can be interfered by others needs to be modeled, then another action *attempt_a* should be added and the permission placed onto the latter. Another alternative is to introduce modalities for trying, as in Santos *et al.* [13].

Lindahl [8] studies *liberty spaces* to present the concept of *less free than*, a relationship between maximally consistent sets of deontic positions. The general idea is somewhat similar to our definition of strictness; however, as Lindahl notes, most of the maximally consistent sets are incomparable using this relationship, whereas our notion of strictness provides interesting theorems.

Many of the above mentioned authors, and also others, deal with some definition of conflicts but they usually leave out the inconsistencies that arise because of the onuses imposed to the other party (see our example of $\mathscr{P}_p(!a)$ conflicting with $\mathscr{O}_p(a)$ in Section 3).

5 Conclusions

In this article we extended our formalisation of contracts for interactive systems [11] to deal with absence of actions, mutually exclusive actions and conflicts. The issues raised by interaction between parties, allowing for collaboration and interference, are particularly interesting in the domain of computer-mediated contracts, in which systems typically work in synchrony and proceed only through handshaked actions. Much work has been done in this domain of synchronous systems from a Computer Science perspective, and we believe that our approach allows us to adopt many existing results into the field of contracts. We are currently applying this approach to the analysis of software requirements documents and studying the classes of rights identified in Kanger *et al.* [7] in an interactive setting.

References

[1] André Arnold (2002): *Nivat's processes and their synchronization*. *Theor. Comput. Sci.* 281, pp. 31–36, doi:10.1016/S0304-3975(02)00006-3.

[2] Maria Fasli (2002): *On Commitments, Roles, and Obligations*. In: *Revised Papers from the Second International Workshop of Central and Eastern Europe on Multi-Agent Systems: From Theory to Practice in Multi-Agent Systems*, CEEMAS '01, Springer-Verlag, pp. 93–102, doi:10.1007/3-540-45941-3_10.

[3] G. Governatori & Z. Milosevic (2005): *Dealing with contract violations: formalism and domain specific language*. In: *EDOC Enterprise Computing Conference, 2005 Ninth IEEE International*, IEEE, pp. 46–57, doi:10.1109/EDOC.2005.13.

[4] Guido Governatori (2005): *Representing business contracts in RuleML*. *Int. J. Cooperative Inf. Syst.* 14(2-3), pp. 181–216, doi:10.1142/S0218843005001092.

[5] H. Herrestad & C. Krogh (1995): *Deontic logic relativised to bearers and counterparties*. *Anniversary Anthology in Computers and Law*, pp. 453–522.

[6] W.N. Hohfeld (1913): *Some fundamental legal conceptions as applied in judicial reasoning*. *Yale Law Journal* 23, p. 16, doi:10.2307/785533.

[7] S. Kanger & H. Kanger (1966): *Rights and parliamentarism*. *Theoria* 32(2), pp. 85–115, doi:10.1111/j.1755-2567.1966.tb00594.x.

[8] L. Lindahl (1977): *Position and change: A study in law and logic*. 112, Springer, doi:10.1007/978-94-010-1202-7.

[9] Olivera Marjanovic & Zoran Milosevic (2001): *Towards Formal Modeling of e-Contracts*. In: *Proceedings of the 5th IEEE International Conference on Enterprise Distributed Object Computing*, EDOC '01, IEEE Computer Society, Washington, DC, USA, pp. 59–, doi:10.1109/EDOC.2001.950423.

[10] Gordon Pace & Fernando Schapachnik (2012): *Contracts for Interacting Two-Party Systems*. Technical Report, FCEyN, Universidad de Buenos Aires. Available at `http://publicaciones.dc.uba.ar/Publications/2012/PS12a`.

[11] Gordon J. Pace & Fernando Schapachnik (2011): *Permissions in Contracts, a Logical Insight*. In: *JURIX*, pp. 140–144, doi:10.3233/978-1-60750-981-3-140.

[12] Y.U. Ryu (1998): *Specification of contractual obligations in formal business communication*. *Data & knowledge engineering* 26(3), pp. 309–326, doi:10.1016/S0169-023X(97)00048-7.

[13] F.A.A. Santos, A.J.I. Jones & J. Carmo (1997): *Action concepts for describing organised interaction*. In: *System Sciences, 1997, Proceedings of the Thirtieth Hawaii International Conference on*, 5, IEEE, pp. 373–382, doi:10.1109/HICSS.1997.10062.

[14] Yao-Hua Tan & Walter Thoen (1998): *A logical model of directed obligations and permissions to support electronic contracting*. *Int. J. Electron. Commerce* 3, pp. 87–104.