Dealing with the Hypothetical in Contracts

Gordon J. Pace

Dept. of Computer Science, University of Malta, Malta
gordon.pace@um.edu.mt

The notion of a contract as an agreement regulating the behaviour of two (or more) parties has long been studied, with most work focusing on the interaction between the contract and the parties. This view limits the analysis of contracts as first-class entities — which can be studied independently of the parties they regulate. Deontic logic [1] has long sought to take a contract-centric view, but has been marred with problems arising from paradoxes and practical oddities [2]. Within the field of computer science, the holy grail of contracts is that of a deontic logic sufficiently expressive to enable reasoning about real-life contracts but sufficiently restricted to avoid paradoxes and to be computationally tractable.

Contract automata [3–5] have been proposed as a way of expressing the expected behaviour of interacting systems, encompassing the deontic notions of obligation, prohibition and permission. For instance, the contract automaton shown in Fig. 1 expresses the contract which states that *'the client is permitted to initialise a service, after which, he or she is obliged to submit valid user credentials and the provider is prohibited from increasing the price of the service.'* Note that the states are tagged with the deontic information, explicitly stating what actions are obliged ($\mathcal{O}$), permitted ($\mathcal{P}$) and forbidden ($\mathcal{F}$) by which party (given in the subscript). The transitions are tagged with actions which when taken by the parties induce a change of state, with $*$ being used as shorthand to denote *anything-else*.

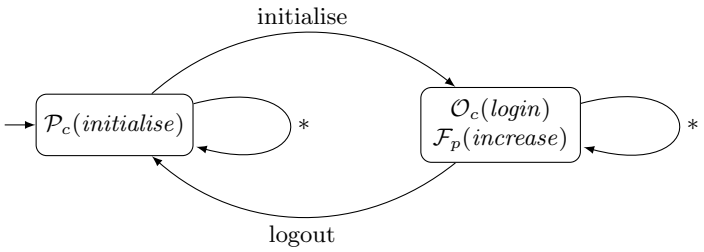

Fig. 1. Service provision contract

The use of automata to describe contracts allows for their composition in a straightforward manner using standard techniques from computer science. For instance, consider a bank client who has a savings account and a loan with the bank. The complete actual contract between the bank and its client is the com-

position of the two contracts regulating the different accounts. This approach thus not only enables the description, analysis and validation of contracts independently of the agents, but also allows for a compositional approach to contract construction.

A limitation inherent to the notion of contract automata is that they describe norms and ideal behaviour only through the knowledge of the agents' actions. No part of a contract may depend on whether or not another part of the contract was violated or not. For example, clauses such as *'if the bank does not give permission to the client to open a new bank account, it is obliged to pay a fine.'* Similarly, it is impossible to specify clauses which depend on whether another clause is also in force, such as *'if p is obliged to pay, he is forbidden from leaving the shop.'* Note the difference between the two cases — whereas the first depends on whether or not a clause is satisfied, the second depends on whether a clause is enforced. We are currently looking into ways of extending the semantics of contract automata to allow for these two kinds of scenarios.

Reparation: Contracts identify ideal behaviour which is not necessarily adhered to — although the contract between the bank and the client states that the client is to ensure that the repayments are done on time, the client does not always do this, in which case another clause would ensure that he or she is now also obliged to pay a fine. These reparatory contract clauses are crucial to allow for the description of behaviour under non-ideal behaviour. This can be handled, for instance, by qualifying transitions also with deontic clauses (or their negation), ensuring that they can only be taken if the clause is satisfied (or violated). The initial state of the automaton shown in Fig. 2(a) shows how this construction is used to ensure that party 1 is permitted to download file A ($downloadA$), as long as party 2 has permission to download file B ($downloadB$)[1].

Although in the example, the tagged clause appears in the source state of the transition, this is not necessarily the case and the approach can be used to reason about clauses *as though they were in force in that state* — hypothetical clauses. For instance, leaving out the $\mathcal{P}_2(downloadB)$ in the initial state of the automaton in Fig. 2(a) would result in a similar contract but in which no violation is tagged when the transition to the next state is taken.

Note that, for a complete semantics one would have to have two types of violations: (i) those violations which took place but have an associated reparation (violating $\mathcal{P}_2(downloadB)$ in the initial state of Fig. 2(a)); and (ii) violations for which there is no associated reparation (violating $\mathcal{P}_1(downloadA)$ in the same state). This allows us to make a distinction between satisfying the contract outright and satisfying but through the taking of reparations.

Conditional clauses: Another extension to contract automata are conditional clauses, based on whether a clause is in force at a particular moment in time e.g. *'if Peter is obliged to pay, then Peter is forbidden from leaving the shop'*,

[1] For completeness we would also want to add which party is satisfying or violating the clause on the transition.

or *'if Peter is not permitted to leave the shop then Peter is prohibited from smoking.'* Fig. 2(b) shows how the first of these examples can be encoded in a contract automaton. Such conditional clauses are particularly useful when composing contract automata, since the presence of particular clauses may not be known before the actual composition.

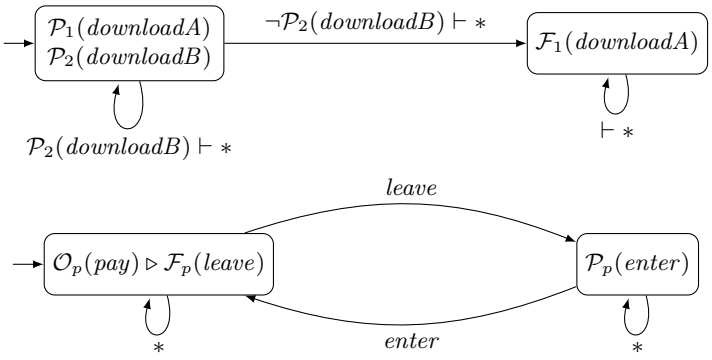

Fig. 2. (a) party 1 is permitted to download file A, as long as party 2 has permission to download file B; (b) if p is obliged to pay, then p is forbidden from leaving the shop.

The interaction of these two forms of deontic tagging can be used to express complex scenarios. For instance, the last example can be extended in such a manner that if Peter violates the prohibition from leaving (when in force), he would be forbidden from entering the shop again. Note that in such a case the transitions may also need to be tagged with conditional deontic clauses.

Furthermore, in these examples we have simply tested for the satisfaction/violation or presence/absence of a single contract clause. Extending this approach to deal with sets of clauses (in a conjunctive or disjunctive manner) or even for whole boolean expressions over these new operators can possibly lead to complex constructiveness issues similar to the ones which arise in Esterel [6]. Even without complex expressions, such cases may arise. For instance, how should the clause $!\mathcal{O}_p(a) \triangleright \mathcal{O}_p(a)$ be interpreted? Does it mean that (i) the obligation to perform a should always be in force even when it is was going not to be the case, or that (ii) it is a void statement which should be disallowed, since if the obligation is not in force, then we add it, but since it is now in force the clause enforcing it no longer triggers and hence cannot be considered to be in force? These and other similar issues make the definition of the formal semantics of such contracts challenging and the development of algorithms for the discovery of conflicts in such contracts non-trivial.

References

1. P. McNamara. Deontic logic. volume 7 of *Handbook of the History of Logic*, pages 197–289. North-Holland Publishing, 2006.
2. J.-J. C. Meyer, F. Dignum, and R. Wieringa. The paradoxes of deontic logic revisited: A computer science perspective (or: Should computer scientists be bothered by the concerns of philosophers?). Technical Report UU-CS-1994-38, Department of Information and Computing Sciences, Utrecht University, 1994.
3. G. J. Pace and F. Schapachnik. Permissions in contracts, a logical insight. In *JURIX*, pages 140–144, 2011.
4. G. J. Pace and F. Schapachnik. Contracts for interacting two-party systems. In *FLACOS 2012: Sixth Workshop on Formal Languages and Analysis of Contract-Oriented Software*, sep 2012.
5. G. J. Pace and F. Schapachnik. Types of rights in two-party systems: A formal analysis. In *JURIX*, 2012.
6. T. Shiple, G. Berry, and H. Touati. Constructive analysis of cyclic circuits. In *European Design and Test Conference*, 1996.