A Controlled Language for the Specification of Contracts

Gordon J. Pace* and Michael Rosner**

University of Malta, Msida MSD2080, Malta
{gordon.pace,mike.rosner}@um.edu.mt

Abstract. Controlled natural languages have been used to enable the direct translation from natural language specifications into a formal description. In this abstract we make a case for such an approach to write contracts, and translating into a temporal deontic logic. Combining both temporal behaviour and deontic behaviour is challenging both from a natural language and a formal logic perspective. We present both a logic and a controlled natural language and outline how the two can be linked.

1 Introduction

Controlled languages are restricted subsets of natural language that have been designed with a specific aim, restricted in vocabulary, syntax and semantics to enable the language to be learned, translated, analysed and generated. Natural language contracts specifying obligations between contracting parties, are so complex that we are generally forced to rely upon costly legal specialists for their formulation and analysing their implications. This motivates a controlled language (cf. Pulman [1]) for contracts, rich enough to be useful in a range of real applications, simple enough for ordinary people to understand, and precise enough to be amenable to automated methods of reasoning.

An important feature of a contract is that one party makes an offer for an arrangement that another accepts. Once this exchange takes place, constraints are enforced over the parties' future actions. Much of the time, contracts are made orally or are implied by the situation at hand. However, beyond a certain level of complexity, written contracts are the norm. The use of unrestricted natural language brings with it a host of problems relating to ambiguity, syntactic complexity, context sensitivity, etc.

Contracts appear in computational settings in various guises — from simple pre- and post-condition pairs in programming giving guarantees on outcomes based on assumptions on the initial conditions, to quality of service agreements requesting, for instance, a maximum guaranteed packet-dropping probability. Service level agreements, contracts used for conformance checking, and system specifications are expressed in a variety of ways, but may still contain interesting deontic nuances. Simple examples of a system specification, combining deontic and temporal notions are given below:

– Upon accepting a job, the system guarantees that the results will be available within an hour unless cancelled in the meantime.

* Dept. Computer Science
** Dept. Artificial Intelligence

- Only the owner of a job has permission to cancel the job.
- The system is forbidden from producing a result if it has been cancelled by the owner.

2 Underlying Logic Representation Language

The challenge to formalise deontic logic, to reason about normative concepts such as obligations and permissions, has been the hot topic of research for various decades. The main challenge is that it is very easy to describe paradoxical situations using deontic concepts [2]. Introducing the concept of time, introduces more paradoxes [3]. Various axiomatizations have been proposed, as an attempt to deal with the paradoxes, however, one of the more effective approaches has been that of restricting the syntax [4]. Since our aim is to reason about contracts derived from natural language texts, and which could thus include paradoxes or contradictions, we opt for a more general logic, which could then be restricted, syntactically or semantically, to weed out potential problems.

Contracts are sometimes seen as properties which should be satisfied by a system. This view however, does not enable (i) reasoning *about* the contract eg 'What are the currently undischarged obligations in the contract?'; or (ii) reasoning about exceptional cases in a contract eg 'Whenever clause (a) is violated, the user is prohibited from obtaining the service'. The introduction of explicit prohibition, obligation and permission clauses into a contract is thus essential to enable reasoning about it. Furthermore, such clauses have to be associated to a particular agent participating in the contract.

The deontic logic includes obligation, permission and prohibition ($O(\alpha : e)$, $P(\alpha : e)$ and $F(\alpha : e)$), non-deterministic choice (+), conjunction (&), conditional ($c_1 \triangleleft \alpha \triangleright c_2$), generalised sequential composition ($c_2 \lll c \ggg c_1$, which starts with c, and then follows it up with c_1 or c_2 depending on whether it was satisfied or violated) and timing information ($c_{[b,e]}$):

$$\begin{aligned} \mathit{contract} ::= {} & \top_{\mathit{time}} \mid \bot_{\mathit{time}} \mid O(\mathit{agent} : \mathit{action}) \mid P(\mathit{agent} : \mathit{action}) \mid F(\mathit{agent} : \mathit{action}) \\ \mid {} & \mathit{contract} + \mathit{contract} \mid \mathit{contract} \;\&\; \mathit{contract} \mid \mathit{contract} \triangleleft \mathit{action} \triangleright \mathit{contract} \\ \mid {} & \mathit{contract} \lll \mathit{contract} \ggg \mathit{contract} \mid \mathit{contract}_{[\mathit{time},\mathit{time}]} \end{aligned}$$

Using these operators and fix-point definitions, other operators can be defined: (i) one branch conditional: $e \rightarrow c \equiv c \triangleleft e \triangleright \top_0$; (ii) sequential composition: $c_1; c_2 \equiv c_2 \lll c_1 \ggg \bot_0$; (iii) the always operator: $\Box(c) \equiv c \;\&\; \top_1; \Box(c)$; and (iv) the sometimes operator: $\Diamond(c) \equiv c + \top_1; \Diamond(c)$. Furthermore, the complement of an action or an agent can be expressed in the logic using a bar over the object. The examples given earlier can be written in the following manner:

- Upon accepting a job, the system guarantees that the results will be available within an hour unless cancelled in the meantime:
  $\Box(accept_j \rightarrow O(system : (result_j + cancel_j))_{[0,1hr]})$
- Only the owner of a job has permission to cancel the job:
  $\Box(P(owner_j : cancel_j) \;\&\; F(\overline{owner}_j : cancel_j))$
- The system is forbidden from producing a result if it has been cancelled by the owner: $\Box(cancel_j \rightarrow F(system : (\Diamond(result_j))))$

The logic proposed shares much in flavour with CL [4] and other action-based deontic logics. One can construct observer formulae (one for each actor), using which one can model-check contracts [5], and perform contract analysis. The temporal side of the logic is based on timed regular expressions [6]. Although, as in timed regular expression, a continuous time domain may be used, at the moment we restrict the time to a discrete domain for analysis techniques. Through the use of a trace semantics of the logic, standard model checking techniques can be used to check for validity.

3 Remarks on a Possible Controlled Language

In this section we take the numbered examples of section 1 and propose some simplifications which reduce both syntactic complexity and, more importantly, the potential for ambiguity.

1. **original:** Upon accepting a job, the system guarantees that the results will be available within an hour unless cancelled in the meantime.
   controlled `if SYSTEM accepts Job, then during one hour it is obligatory that SYSTEM make available results of Job unless SOMEONE cancels Job.`
   comment: There are three events: accepting a job, results being available, and a cancellation. There is also a contractual obligation concerning the second event, but this is discharged if the cancellation takes place. This sentence displays the classical problem of attachment ambiguity. There is also a problem of ellipsis. The main problems are (a) the attachment of the time adverbial *within an hour* and (b) the object of the the cancellation. Regarding the attachment, the adverbial could attach to either the availability of the results, or to the obligation concerning the availability of results (cf. *within an hour I promise to go* vs. *I promise to go within an hour*). Regarding the object of the cancellation, this could be the job or the results. We should note that in section 2, both these ambiguities have been resolved: attachment is to the obligation has been favoured, whilst the object of the results is assumed to be the results.
   Several problems can be solved by allowing proper names into the language. Predefined ones, like `SYSTEM`, are notated using all capital letters, whilst arbitrary ones, like `Job`, are used to enforce coreference, have an initial capital letter. A second major change is a rationalised event syntax based on a simple agent-action-object format. The easy cases are underlined above. The complex case revolves around `is obliged`, which can usefully be treated as an event (strictly it is a state) whose object is itself an event. We have carefully controlled the syntax and placement of the time adverbials, and, last but not least, the position of `unless` is immediately after the statement of the obligation.
2. **original:** Only the owner of a job has permission to cancel the job.
   controlled: `it is permitted that only owner of Job cancels Job.`
   comment: The syntax is basically uncontroversial but there is an issue about the anaphoric status of definite noun phrases. Although it would be possible, following Fuchs-et-al. [7], to deal with this using DRT ((cf. Kamp and Reyle [8]),

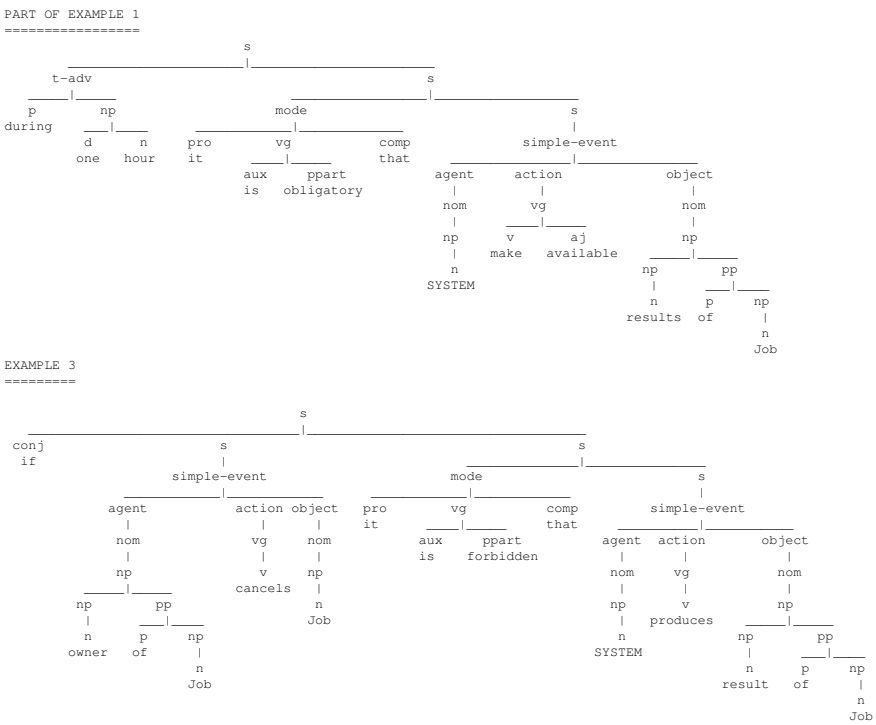
we feel that the introduction of proper names offers a much simpler solution. The logical representation suggested in section 2 is being driven by the word `only` is straightforward to engineer using techniques from unification grammar.

3. **original:** The system is forbidden from producing a result if it has been cancelled by the owner.

   controlled `If owner of Job cancels Job, it is forbidden that SYSTEM produces result of Job`

   comment: The main problem with the original sentence is that it the syntax is complex, and the word *it* ambiguously refers to either *the system* or *result* or *producing a result"*, or, if we consider all three sentences to be part of a discourse, the job mentioned in the second sentence. Although it has been argued that this particular case could be resolved by semantic constraints or or by the use of heuristics (cf. ACE [7]), we feel that all these problems can be avoided by adopting the controlled language being suggested.

Space limitations prevent a proper discussion of the grammar. Instead we illustrate some proposed parse trees for key phrases in our examples.



The examples above have been generated using a PC-PATR, a version of PATR2 (Shieber [9]). Unification grammar formalisms such as PATR2 facilitate the transformation of parse tree fragments shown into attribute-value structures that can be regarded as notational variants of the formulae presented in section 2 of this paper.

4 Conclusions

In this abstract we have outlined the proposal for a controlled natural language and a deontic logic to enable specification and reasoning about contracts in an automatic fashion. A number of challenges still remain to be addressed.

In the translation from the sublanguage to logic, one can find a direct correspondence between the subgrammar and the logic. We plan to use contract analysis techniques similar to ones we have recently developed for CL to enable contract sanity checking, including discovery of conflict analysis and superfluous clauses. These techniques can provide feedback to the contract translation from the natural sublanguage into logic, for example, for the resolution of ambiguities.

The main aim of transforming logic to language is to make it more understandable to those unfamiliar with logic. The transformation is clearly a problem of Natural Language Generation (NLG) which is generally recognised (cf. Reiter and Dale [10]) to involve large numbers of realisation choices. These can of course be by-passed by adopting a suitably strict generation algorithm. The extent to which this compromises naturalness is a factor that will need to be carefully evaluated in the contracts domain.

References

1. Pulman, S.: Controlled language for knowledge representation. In: Proceedings of the First International Workshop on Controlled Language Applications, Leuven, Belgium (1996)
2. Meyer, J.J.C., Dignum, F., Wieringa, R.: 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. Pace, G.J., Schneider, G.: Challenges in the specification of full contracts. In: Proceedings of Integrated Formal Methods (iFM'09). LNCS, Springer (2009)
4. Prisacariu, C., Schneider, G.: A formal language for electronic contracts. In: 9th IFIP International Conference on Formal Methods for Open Object-Based Distributed Systems (FMOODS'07). Volume 4468 of LNCS., Springer (2007)
5. Pace, G., Prisacariu, C., Schneider, G.: Model checking contracts –a case study. In: 5th International Symposium on Automated Technology for Verification and Analysis (ATVA'07). Volume 4762 of LNCS., Tokyo, Japan, Springer-Verlag (2007)
6. Asarin, E., Caspi, P., Maler, O.: Timed regular expressions. Journal of the ACM **49**(2) (2002) 172–206
7. Fuchs, N.E., Kaljurand, K., Kuhn, T.: Attempto Controlled English for Knowledge Representation. In Baroglio, C., Bonatti, P.A., Małuszyński, J., Marchiori, M., Polleres, A., Schaffert, S., eds.: Reasoning Web, Fourth International Summer School 2008. Number 5224 in Lecture Notes in Computer Science, Springer (2008) 104–124
8. Kamp, H., Reyle, U.: From Discourse to Logic: Introduction to Model-theoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory. Studies in Linguistics and Philosophy. Springer (1993)
9. Shieber, S.: An Introduction to Unification-Based Approaches to Grammar. CSLI Publications, Stanford University, California (1986)
10. Reiter, E., Dale, R.: Building natural language generation systems. Studies in natural language processing. Cambridge University Press, Cambridge, U.K. ; New York (2000)