Implementing Responsibility for States and Events

Martin J. Kollingbaum
Dept of Computing Science,
University of Aberdeen,
Aberdeen, AB24 3UE, U.K.
mkolling@csd.abdn.ac.uk

Timothy J. Norman
Dept of Computing Science,
University of Aberdeen,
Aberdeen, AB24 3UE, U.K.
tnorman@csd.abdn.ac.uk

Chris Reed
Dept of Applied Computing,
University of Dundee
Dundee, DD1 4HN, U.K.
chris@computing.dundee.ac.uk

ABSTRACT

Contracts in the real world often rest upon a notion of responsibility, by which parties commit to the fulfilment of particular imperatives embedded in the contract. Responsibility is not the same as direct action, nor commitment to such action: a canonical example is where imperatives are issued, in a particular context, to effect the delegation of responsibility. Furthermore, responsibility can range not only over particular activities, but also over particular states of the world. This paper first explains the problem of state-based and event-based responsibility, and then illustrates how this is operationalised in the NoA system through the use of an example.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems, Coherence and coordination*

General Terms

Agent Architecture

1. INTRODUCTION

Contracts form a fundamental component of many multi-agent systems, whether as explicit objects, or as implicit agreements about adherence to the rules and norms that govern interaction. Accurate, robust, and flexible representation of contracts is thus an important requirement as multi-agent systems become larger and more complex.

A contract typically captures a set of responsibilities of some number of agents: the responsibility that one agent has of delivering goods to another; the responsibility that an agent in a position of power has to ensure that its subordinates meet some deadline; the responsibility that an agent has should it default upon its agreements, etc.

The notion of *responsibility*, however, has not had a thorough treatment in recent research. Models of commitment [1] have been developed to represent the binding of an agent or group of agents to a particular course of action, but this fails to account for an agent taking on responsibility without direct action being required. Models of normative systems [4] succinctly capture what an agent must (not) do, but not what it is responsible for [2]. In order to build multi-agent systems in which contextual features and organisational structures support mechanisms for the transfer of responsibility – which are a feature of complex human societies – a better understanding of responsibility is required. Further, any model of responsibility must account for the fact that it may range of both activities and states of the world [2, 5]. Here, we offer a motivation for this bipartite approach to responsibility by briefly summarising some of these arguments and discussing an implemented agent architecture: NoA.

AAMAS'03, July 14-18, 2003, Melbourne, Australia.

2. STATES AND EVENTS

Consider a simple delivery domain where trucks, `t1` and `t2`, are managed by agent `m`. The manager is responsible for the fulfilment of delivery contracts, and will do so through the issuing of instructions to the drivers of these trucks. Consider the following two instructions: (i) Make sure you get to `c` by 5pm! (issued to the driver of truck `t1`) (ii) Use route `r1`! (issued to the driver of truck `t2`). The first refers to a state of affairs that is to be achieved, and the second to the performance of a specific action.

Consider equating the second statement to the state of affairs in which getting from `a` to `b` via route `r1` has just been done: the state `done(move(t2, a, b, r1))` (see fig 1). In this way, a logic of agentative action need only capture the achievement of states of affairs, some of which are special "done" states. This is philosophically and pragmatically problematic. From a philosophical perspective, the performance of actions is captured through the use of special pseudo-states (as Hamblin [2] disparagingly refers to them) that simply serve to record the sequence of actions that have been performed in order to get to the present state. Two apparently identical states are different precisely because of the actions that have been performed. This approach is problematic from a practical point of view for two reasons. First, practical problems often become manifest at the logical level when it is essential to keep track of "done" events in every state. Second, actions are typically specified as having a set of preconditions and a set of effects. Suppose that an agent is motivated to achieve the state `done(move(t2, a, b, r1))`. This state is not an effect of the action `move(t2, a, b, r1)` (see fig 1). The decision-making mechanism of the agent must, therefore, treat such "goals" in a different

way to, for example, achieving `in(t2,b)`.

Rather than assuming states to be primary, an alternative is to see events as primary. On this view, a state is then simply the pre- or post-condition of an event. But again, problems arise. In particular, if a state is simply the result of a series of events, it becomes difficult to account for the distinction between successful and unsuccessful action, without substantial additional theoretical machinery. Secondly, in reasoning about how to perform sequences of events, the notion of a side-effect becomes conflated with other event effects, reducing expressivity (an argument well rehearsed in work on causal link planning).

3. IMPLEMENTATION IN NOA

The behaviour of a NoA agent is motivated by norms: obligations to be responsible for the achievement of states of affairs and for the performance of actions. In the NoA language this distinction is represented through the use of `achieve` and `perform` statements respectively. These activities then have the potential to influence the behaviour of the agent through the generation of instantiated plans that are options for execution. These options are then filtered on the basis of the states that an agent is permitted to achieve and actions that it may perform. Plan options that have side-effects that conflict with the agent's norms or that represent actions that are forbidden will be removed from the set of options. One of the remaining options is then selected for execution. All norms specify the agent involved, the state of affairs or action concerned, the condition that activates the norm (e.g. `lic(X,c)`) and the condition that deactivates the norm (e.g. `not lic(X,c)`) (fig 1: `2`).

Following the examples used in the previous section, consider a simple map of three locations, `a`, `b`, `c`, connected by directed routes `r1` (from `a` to `b`), `r2` (`b`–`a`), `r3` (`a`–`c`), `r4` (`a`–`c`), `r5` (`c`–`b`) and `r6` (`b`–`c`). The map includes multiple routes between given locations, and restrictions on routes, which correspond to real world phenomena such as tolls, weight limitations or license agreements. Here, trucks are normally prohibited from entering location `c` and from using route `r1` (fig 1: `0`, `1`), but the manager has obtained licences that allow it to be responsible for truck `t1` to enter location `c` and `t2` to use route `r1`. In general, whenever the `lic` fluent holds for an agent `X`, agent `m` is permitted to bring it about that `X` is in location `c`, up until such time as the fluent no longer holds (fig 1). These licences provide them with specific permissions, which override the general prohibitions (fig 1: `2`, `3`).

There are several points to note about the domain. Vehicles can be present at locations: the states of the system. The movement of vehicles from one location to another correspond to events. The graph is directional: a route from `a` to `c` does not entail a route from `c` to `a`. There are multiple routes between given locations (routes `r4` and `r5` from `a` to `c`). An event is thus not uniquely identified with a change from one state to another. There are any number of routes leading to a given location, that is, a given state does not equate to the immediately recent performance of any given event. Both routes and locations can be individually restricted, without affecting the locations they are connected to, or the routes by which they can be reached.

With these structures in place, it is easy to run through a sample interaction in the domain. Suppose the manager, `m`, agrees to fulfil a contract in which a parcel `p1` is transported

```
0 prohibition(m, achieve in(X,c), T, F)
1 prohibition(m, perform move(X,a,b,r1), T, F)
2 permission(m, achieve in(X,c), lic(X,c), not lic(X,c))
3 permission(m, perform move(X,a,b,r1),
                lic(X,r1), not lic(X,r1))
4 lic(t1,c)   lic(t2,r1)
5 plan move (X,A,B,R)
6   preconditions (in(X,A), road(A,B,R))
7   effects       (in(X,B), not in(X,A)) { ... }
8 obligation(m, achieve deliv(p1,c), T, deliv(p1,c))
9 obligation(m, achieve deliv(p2,b), T, deliv(p2,b))
```

Figure 1: A simple specification in NoA.

from location `a` to location `c` and another parcel `p2` from `a` to `b`. The process by which such contracts are agreed is beyond the scope of this paper (but see [5] for a formal account, and [3] for an implementation). The result of the contract negotiations are normative states of affairs (fig 1: `8`, `9`), the initial state containing `in(p1,a)` and `in(p2,a)`. In the following discussion, we will focus on the transportation of these parcels, and assume that the manager agent has plans that expand to loading the parcel on a truck, achieving `in(Truck, A)` and unloading the truck.

Let us consider first the delivery of parcel `p1`. There are four possible ways to achieve this state with the `move` plan: either `t1` or `t2` using route `r3` or `r4`. However, those instantiations involving `t2` transform the world into a state of affairs that is explicitly prohibited: `in(t2,c)`. Only the options involving the licensed truck `t1` remain possibilities (they are not pruned by the filtering process). Similarly, parcel `p2` may be delivered by either truck: `t2` delivering the parcel using route `r1` or by `t1` continuing from `c` to `b` via route `r5` (`t1` using route `r1` is not a permitted action).

Note that for both goals the filtering process prohibits the use of a plan not because a state of affairs or an event is prohibited directly, but rather that `m` is prohibited from being responsible for bringing about that state of affairs. This is a powerful distinction which supports reasoning about responsibility along long or complex chains of delegation in organisational structures.

4. CONCLUSION

In order to capture the responsibilities of an agent, as distinct from the commitments it maintains and the norms by which it functions, it is necessary to distinguish explicitly between states and events. A formal, modal system has been summarised elsewhere [5] that provides a solid foundation for understanding responsibility, and has been used in implementing a working agent architecture. With future research focusing on implementing theoretical work in delegation and group responsibility, the approach promises to be a powerful tool in implementing solutions in rich domains.

5. REFERENCES

[1] P. R. Cohen and H. J. Levesque. Teamwork. *Noûs*, 25(4):487–512, 1991.

[2] C. L. Hamblin. *Imperatives*. Basil Blackwell, 1987.

[3] M. J. Kollingbaum and T. J. Norman. Supervised interaction. In *AAMAS-02*, pp. 272–279, 2002.

[4] J.-J. C. Meyer and R. J. Wieringa, editors. *Deontic Logic in Computer Science*. Wiley, 1993.

[5] T. J. Norman and C. A. Reed. Delegation and responsibility. *Intelligent Agents VII*, pp. 136–149, 2001.