An Expectation Framework for Agent Social Reasoning

Iain Wallace
PhD Student
Centre for Intelligent Systems and their Applications
School of Informatics
The University of Edinburgh
i.a.wallace@sms.ed.ac.uk

Abstract

In multi-agent systems it is necessary for agents to reason not only about their actions, but also about their interactions with others. I hypothesise that there are benefits to separating social from practical reasoning systems such as BDI, and propose a possible solution.

1 Introduction

Social reasoning refers to how an agent should deliberate about interactions with other agents. This is related to practical reasoning, which is concerned with an agent's general rational behaviour. Although the integration of specific social aspects with practical reasoning has been studied before, there has been less emphasis on social reasoning in the general case. Practical reasoning has the Belief-Desire-Intention (BDI [1]) theory to champion as a general approach - there is no such general approach to social reasoning so the overall viewpoint is lost. Consequently, it is difficult to compare many approaches to the different facets of the social reasoning problem. Part of the reason for this is that the interaction of social and practical reasoning presents problems. Often previous social reasoning approaches work within the constraints of BDI [5], but this can be complex, as it was not designed to explicitly consider social issues. As such, some concepts can be impossible to represent without specific extensions.

The weaknesses in existing approaches to social reasoning lead me to the following thesis:

> Would a general method of social reasoning, separate from practical reasoning, allow for the design of simpler or more powerful agents?

I intend to develop a logical framework for practical social reasoning. Then from this create a reasoning architecture, and even implementation. To answer the question an evaluation of the relative simplicity and expressiveness of agent implementations based on this and traditional agent designs would be carried out. This framework could then be used for three purposes:

- Adding dynamic social reasoning to account for change in open systems - currently most social reasoning is implicit in the agent's structure and carried out by the designer ([4],[3]).
- Comparison of existing agent reasoning architectures. Currently this is difficult, as many approaches focus only on one facet of social reasoning, and different approaches may be at differing levels of abstraction.
- Investigation of the relation between practical and social reasoning. Considering social reasoning as a separate, though linked, process may enable the design of more powerful social agents.

The Expectation–Strategy–Behaviour model (ESB, Rovatsos [7]) is an attempt to specify such a framework, separate to general practical reasoning. An agent's actions are governed not only by its goals - as in BDI agents - but by expectations of other agent's behaviour. Expectations are adapted based on experience and observation,

as in open dynamic environments we cannot be sure every other agent is entirely co-operative and reliable. It should be noted that despite recognising that BDI is the most prevalent practical reasoning theory I do not necessarily assume that ESB will be paired with BDI - though it is likely future work may take advantage of such a restriction.

2 The ESB Theory

Any social reasoning must involve modelling hidden variables of other agents - for example the internal states of other agents and the relationships between them (e.g. in the learning automata of [2]). This is necessary for an agent to predict the outcome of any social interaction and choose its own actions. The ESB approach provides a processing mechanism for an agent's assumptions about hidden variables, and a method to update these assumptions based on experience.

At the heart of the ESB method is what it means for an agent to have an *expectation*. The expectation "whenever condition C holds, agent a will expect event E, verify it with ϕ and react with ρ^+ if it is true or ρ^- otherwise" is represented through a 6-tuple as below:

$$(EXP\ a\ C\ E\ \phi\ \rho^+\ \rho^-) \tag{1}$$

EXP – The *name* of the expectation.

a – The *agent* holding this expectation.

C – The *condition* under which the expectation holds. This has to be observable from the point of view of agent a, though could be a condition on the agent's internal state.

E – The expected '*event*'. This could be an event such as "agent b pays a" or it could be an expectation that another agent is assumed to hold. Nesting expectations allows modelling of other agent's mental states, and adaptive behaviour on the part of either agent.

ϕ – The *test* which confirms (or rejects) the expectation of E.

ρ^+ **and** ρ^- – are the positive and negative *response* to the test ϕ. These could be actions the agent should take or modifications to its internal state, including modifying the set of expectations it holds in some way. For example, a trust measure could be incremented on completion of a successful trade.

The other components of ESB are strategies and behaviours. Based on its expectations an agent's strategy takes account of potential behaviour of themselves and others, and how action choices affect future expectations. Then from a valid strategy appropriate social behaviours can be applied to guide the agent's interaction.

2.1 Strategies

A strategy is the mechanism for accounting for how interactions may affect future behaviour. From the current expectations a graph can be generated where each node is a possible future expectation set according to changes possible from the current expectation set's responses, as illustrated in Figure 1. Of course for the initial vertices generated only the responses attributed to the currently valid expectations apply, but for reasoning further all expectations must be considered as future environmental conditions cannot be predicted. This tree represents the social strategy of the agent - how it will manage the social belief rules on which its interaction is based. Dynamic update of these social rules can be compared to belief revision methods, except here these changes may affect not only the agent's own rules but also those nested rules about others. The responses are the update mechanism itself, however these are not limited to updates but may also include external actions.

2.2 Behaviours

Given the strategy graph, a behaviour is a method to generate actions from it - these could be either internal state modifications or influence the practical reasoner directly. For example, a possible behaviour could be to perform an action if it is expected to lead to a trust increase. On the strategy graph in Figure 1 it may be that $C_2 = do\_trade(a, b)$ and $E_2 = high\_trust(a, b)$. ρ_2^+ could add an expectation based on this trust, and ρ_2^- could remove $EXP1$ which might depend on it. This would give behaviour such as offering to trade with b, as that would lead to the

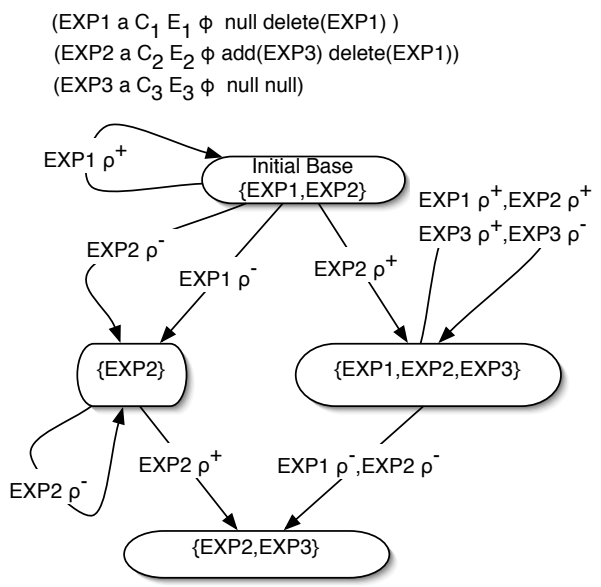

Figure 1: Example Strategy Tree

expected event E of a greater trust rating. An important point is that it's only *expected* that trust will increase when there's a trade between two agents - the test would be if a trade concludes successfully.

3 Defining a Framework

Developing a logic-based framework for ESB will have two main uses: deriving proofs about reasoners, agents and architectures; and guiding the design of a reasoner. By defining the semantics for all parts of an expectation logically, the scope of an agent's reasoning will be bounded. What follows are my current thoughts on what this framework might look like.

3.1 Expectations

The key components of an expectation have different purposes, so the representation will require different properties and restrictions:

C – Likely to be a first-order logic statement, with axioms defined on the other parts of an agent's structure to ensure that it's an observable property (e.g. by restricting it to predicates which are in the agent's beliefs).

E – Should be a hidden property, and restricted in the opposite way to C. Any observable term in E could either be moved into C or ϕ. In addition the language used for E should be extended to include expectations themselves, for recursive modelling.

ϕ – This should be observable and as expressive as C - in both cases they should be decidable statements. There could possibly be some stronger restriction here, as any test must be of a property that follows from E - as it is the test that E occurred. However this is unlikely to be practical, and this reasoning likely to be done by an agent designer.

ρ^+ **and** ρ^- – May be not only acting on the practical reasoner, but changes to internal state and expectations. It is likely for practical use that a language of responses including making specific changes to expectations - e.g *remove* or *change*ϕ - would be defined.

These properties will be refined, with more precise definitions of the predicates and axioms. Even in the loose form above they provide clues as to the shape of a reasoner. As a first step C and ϕ are likely to be restricted to propositional logic. Extending them to first order logic to allow quantifiers would increase expressiveness, but may not always be needed, so the simpler approach could work and ensure decidability. The language used for E is likely to depend a lot on the implementation of behaviours, as it is these which use E directly. It may be that E is simply a subset of the agent's possible beliefs - in a system with no nested modelling. The restriction on E that it is a hidden property may need to be revised, as it is possible to foresee examples of events the agent would like to capture in an expectation rule that are purely observable, and so would have the details pushed into C and ϕ, making E simply *true* - and no help to the reasoning process.

3.2 Strategies

As a strategy is a graph that follows from the current expectation set and extends into the future, it seems likely that logically a strategy could be defined as a model similar to those in computational tree logic (CTL [6]). This would allow conditions for behaviours to be defined with

CTL-like statements - provable properties of the expectation graph which would trigger certain social behaviours as discussed in section 3.3 below. The generation of this model will be linked to the descriptions of expectations - specifically the responses which are applied to the initial base step by step to create the tree.

The key consideration in any implementation or reasoner design will be the method by which the tree is generated. Apart from the definition of each node - a single expectation, or the complete set - there is the problem of how far to build the tree. As the initial expectations are known it may be possible to pre-generate the complete graph, and just keep track of the agent's current position in it. However, it is thought that certain restrictions on the representation will be required to ensure this is the case.

As well as its direct use for agent reasoning, study of an agent design's strategy graph could provide new insights or allow comparison of the properties of different social schemes represented in the more general ESB form. Study of the graph could reveal convergence toward a single set of expectations, loops in the graph may indicate inconsistency in the design, or several other properties. It is thought that further definition of the logic required will enable further consideration of specific graph properties and their meaning in agent reasoning terms.

3.3 Behaviour

Treating the strategy tree as the model for a CTL style logic would allow a CTL-like logic for asserting pre-conditions on behaviours. For example "AFp" which in CTL has the meaning "it is inevitable that p" could have a similar meaning for ESB. One can imagine a behaviour with a condition such as "$AF\ trusts(b)$" to mean "it is inevitable that I will hold the expectation I trust b". The behaviour could then assert a social policy such as to interact as if b was already trusted - even although this is not the case, the agent has reasoned that it is inevitable given the current conditions.

In terms of a reasoner, behaviours are likely to be similar to the plans of BDI implementations, which sequences of actions or goals with guard pre-conditons. A designer will know the range of social actions available to an agent, and merely specify the pre-conditions of each social plan in terms of properties that follow from the tree.

4 Conclusions

This paper presents only a brief overview of my proposed approach to separating agent social reasoning from practical reasoning. The next step towards developing these ideas further will be refining the ideas on the logical representation of expectations to create the theoretical framework. Currently only the rough requirements outlined above exist, these will be fleshed out and logical theories and axioms defined. Once this framework exists, case studies of existing social reasoning problems will pinpoint weakness before the architecture for a reasoner is developed. It is hoped that the eventual goal of a more general social reasoning theory will be of benefit to the agents research community at large, allowing more principled discussion of the issues involved in open MAS.

References

[1] M.E. Bratman, D.J. Israel, and M.E. Pollack. Plans and resource-bounded practical reasoning. *Comput. Intell. (Canada)*, 4(4):349 – 55, 1988.

[2] D. Carmel and S. Markovitch. Learning models of intelligent agents. *NCAI-96*, vol.1:62 – 7, 1996.

[3] Phillip R. Cohen and Hector J. Levesque. Teamwork. *Nous*, 25(4), 1991.

[4] K. Decker and V. Lesser. Designing a Family of Coordination Algorithms. *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, pages 73–80, 1995.

[5] F. Dignum, D. Kinny, and L. Sonenberg. From Desires, Obligations and Norms to Goals. *Cognitive Science Quarterly*, 2(3-4):407–430, 2002.

[6] Dov M. Gabbay, Mark A. Reynolds, and Marcello Finger. *Temporal Logic: Mathematical Foundations and Computational Aspects*, volume 2. Oxford Science Publications, 2000.

[7] Michael Rovatsos. Practical social reasoning systems. *unpublished*. http://homepages.inf.ed.ac.uk/mrovatso/papers/rovatsos-esb.pdf.