Logic, Reasoning and A Programming Language for Simulating Economic and Business Processes with Artificially Intelligent Agents

Bruce Edmonds, Scott Moss and Steve Wallis,
{b.edmonds, s.moss, s.wallis}@mmu.ac.uk**,**
Centre for Policy Modelling
Manchester Metropolitan University
Aytoun Building
Manchester M1 3GH, UK

Abstract

The merits of modelling within a logical, as opposed to Bayesian, framework is discussed. It is claimed that a logical formalism is more appropriate for modelling qualitative decisions and that this framework makes the unfolding of process more apparent. This difference in approach leads to adopting a declarative programming rather than imperative paradigm. This approach also enables the credible modelling of agents with limited information processing capacities. An agent orientated and strictly declarative computer modelling language is presented called SDML which has been specifically developed to support such a style of modelling. Some methodological issues arising from this are also discussed.

1 Introduction

Decision theorists rely on Bayesian processes in order to have a rigorous and clear basis for descriptions of learning and expectations formation and calculations of optimal decisions. However the assumptions of Bayesian theory are actually highly restrictive. It is necessary to update the probabilities of the exhaustive set of mutually exclusive assumptions. Often, this is in the form of an hypothesis being true with probability p or false with probability $(1 - p)$. If there are more competing hypotheses, there can be no overlap among them. Secondly, the various items of evidence used must be mutually independent. The observation of evidence e_i and the observation of evidence e_j conditional upon hypothesis H must be the product of the conditional probabilities rather than the product of the probabilities plus the probability of both e_i and e_j. While it is always possible to choose a language of observations for which this is the case, when there are more than two observations of evidence and the various items are not independent of one another, then Bayes Law rapidly becomes intractable. For example, in most situations where learning is taking place, you would need to frequently update this language but many applications of Bayesian theory assume that this language is constant in order to retain such tractability.

This is simply one manifestation of a problem that runs right through conventional economic modelling. Uncertainty is represented as some kind of probability distribution — often asserted to be subjective. These probability distributions imply that the agents act as if they believe they know everything that can occur to them conditional upon any action that they might take. If behaviour is represented as a process of constrained optimization, then, in addition agents must act as if they know every possible binding constraint and its effect (*e.g.* its shadow price).

These representations imply the assumption that information is limited relative to the information-processing and computational capacities of agents. Economic modellers and their ilk frequently claim that the assumptions are made for simplicity. This may have been acceptable in the days when modellers were laminated to analytically tractable equations due to lack of greater computational resources. Now with the advent of such modelling tools as those described here this is no longer the case and hence we do not understand why such simplicity is an overriding virtue. Now it is feasible, we think that modellers should make the best possible estimate of the cost of that simplicity in terms of relevance and accuracy of the inferences drawn from their models.

Both Bayesian and constrained-optimization representations of behaviour can be avoided by relying on some techniques derived from the field of artificial intelligence. The common attribute of many artificial intelligence techniques is that they entail a recognition that information-processing and computational capacities are limited so that exhaustive searches of the action or strategy space would take longer than the time available for completing an action or that the opportunity cost would be prohibitive. But, as with economic modellers, there is a widespread and deep concern for rigour amongst artificial intelligence scientists. Indeed, one reason artificial intelligence scientists develop and use formal logics is in order to have a rigorous and clear basis for descriptions of intelligent reasoning and implementations of artificially intelligent reasoning. These logical formalisms can provide a solid basis for modelling economic and business processes with intelligent agents.

2 Modelling Qualitative Information

It has been recognized for some time that a difference between formal logics and Bayesian methods is that the former deals more naturally with qualitative information and with limitations on the abilities of decision-makers to identify all possible outcomes or even completely to specify their own preference functions[1].

This is because formal logical languages can also include a universe of user-defined objects, functions and predicates which can directly represent such information in a natural way. Thus the formal logic can be easily extended with the addition of such terms in a very flexible and expressive way. The formal structure of the logic is thus the "glue" that is used to relate these, in a similar manner that Bayesian theory relates probabilities. Perhaps the ultimate indication of the expressivity of a logical formalism is that it can be used to formalise Bayesian theory fairly easily, where this is not possible the other way around.

3 Exploring Decision Making Processes

A point which is not emphasized in the literature is that logics are used more naturally for the analysis of processes while Bayesian and similar methods are used more naturally for the analysis of states.

The reason is that logics support the inference of true statements based on other inferences and axioms (which are also true statements). Thus these latter inferences are only implicitly encoded by the initial statements and the result only revealed to us explicitly (since we are, ourselves, beings of limited rationality) when the consequences of those statements in the logic are worked out. On the other hand, Bayesian processes state what adaptations shall be anticipated or, based on those anticipations, what actions shall be taken.

The true statements (axioms and inferences) naturally describe states so that the inferences from them are naturally taken to describe responses to those states. These responses are what we think of as processes. Bayesian models explicitly describe actions and the results of those actions (effectively processes) to yield what we think of as states. Thus, to a large extent a *process* modelled in a Bayesian way can not be a surprise to us, since it is specified more directly.

---

[1]See, for example, Cohen (1985) or Fox, Clark, Glowinski and O'Neil (1990).

4 Declarative and Imperative Computing Paradigms

These differences correspond to differences in computer programming paradigms. Declarative programming languages such as Prolog are rather like logics in that programs declare statements that are known to be true and relationships between these and other statements. For example, true statements can be asserted to databases, and the inference machinery of the programming language can then “animate” this by making deductions about the truth of other statements. Given an initial state, the rules of the system together with the language’s inference mechanism determine the process that will unfold. The programmer can be surprised at this process as well as about the unfolding results.

Imperative programming languages such as Fortran, C or Pascal state what shall be done in given conditions. They start with an initial state and an *explicit* set of instructions that describe the process that will unfold. Thus there is no possibility here of any surprise about this process, as it was already determined by the programmer. What can be a surprise (and thus informative) is the states that it reaches. In conventional economic modelling, for example, utility or profit maximization are often the representations of the behavioural processes of agents. Since the actual processes are not represented, the models relate outcomes to initial states in a predetermined manner. While the outcomes might surprise the programmer, the processes cannot since they are either not directly represented at all or their outcomes are consequent only upon the initial conditions of the system.

It is easier and more natural to implement logic-based programs in declarative languages but easier and more natural to implement programs incorporating Bayesian processes in imperative languages. This is not to say that imperative programming approaches can’t be used for logical inference or to examine an unfolding process or that declarative programming approaches cannot be used to calculate imperative programmes of action. It is just each is more suited to a particular task, in terms of ease of programming, interpretation, speed of execution and debugging.

You choose your formalism and hence programming language depending on whether you are interested in final states reached or processes independently of whether or not a final (equilibrium) state is reached.

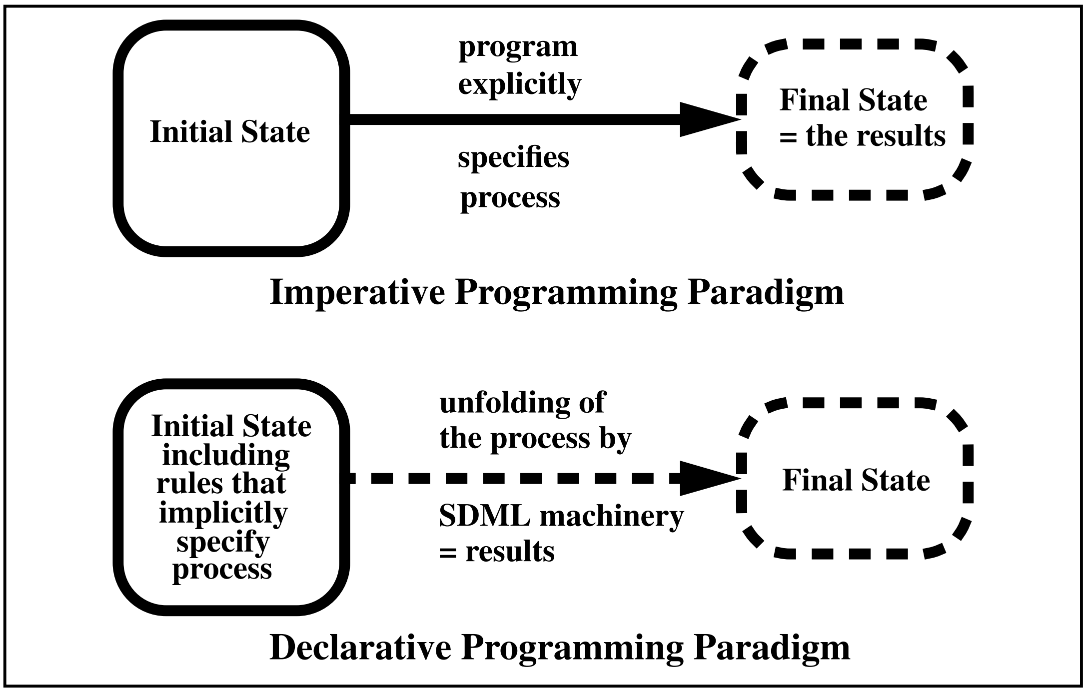

Figure 1: Imperative vs. Declarative paradigms

5 Modelling Agents with Limited Rationality

A second issue is how agents adapt their knowledge and information in a world where Bayesian methods are inappropriate. A natural approach to dealing with this issue is to assume that agents *reason* about their environments and seek to identify the reasons why they perform well or badly in those different circumstances.

The use of declarative programming languages enables us to model this reasoning process when agents have limited computational and information-processing capacities. This is because instead of modelling only numerically defined conditions and outcomes, we are given a suitably expressive language in which to model the computational, learning and decision-making processes of agents. Credible internal states, which represent the structure of knowledge in relation to its environment, can be specified and modified as the agent develops. Since declarative programming languages are also easier to relate to formal logics, they enhance the prospects for developing rigorous models of processes involving intelligent agent behaviour.

If such models are not subsets of known formal logics, it might at least be possible to identify the logical properties which such models lack and then to decide whether the missing properties are important or even appropriately absent.

6 A Strictly Declarative Modelling Language (SDML)

At the Centre for Policy Modelling we have developed a language called SDML (Strictly Declarative Modelling Language) which is strictly declarative in that it has no imperative elements and which represents agents and their environments as collections of rulebases and databases.

Each agent has its own set of databases and rulebases to support the modelling of its decision making processes. Roughly speaking the databases record the facts pertaining to the agent or situation (including explicit representation of its knowledge and beliefs) and the rulebases hold the rules with which new inferences (on the database) can be made. Thus each agent has a rulebase for reasoning about its own behaviour and for learning in the sense described by Edmonds and Moss [3]. Each agent has two such rulebases: one of these determines specific actions to be taken in given conditions and an other, a meta-level rulebase, which can write the rules which are contained in the action-determining rulebase. With this architecture, agents can reason about their own rules of action.

The language can be used for modelling both synchronous and non-synchronous processes and has an interface particularly suited to the modelling of behaviour in organizations, markets and whole economic systems. A large collection of similar agents can be implemented as easily as one, with each agent inheriting common rules to determine its behaviour according to its type. Agents also have databases specific to each date, enabling the 'knowledge' they represent to develop over time[2].

The language has a sophisticated interface (derived from that of its base language - Smalltalk) that allows the user to browse the contents of any rulebase or database at any time, or collate the results in many such databases. One view of this is shown in figure 2.

Figure 2: An example of SDML's interface in use

More details of the workings of SDML can be found in [7].

[2] Their action-deciding object-level rules can similarly develop, allowing their behaviour to radically change over time according to their decision making purposes.

7 Formal Logic and SDML

As SDML is a declarative language it is already close to being a logic. Rules are stated as logical relationships between facts such that if the antecedents are true so are the consequents. These rules are specified beforehand by the modeller to determine the general actions of each agent in the model[3]. Once the model rules and starting facts are established by the modeller, the inference machinery of SDML animates the model by determining the logical consequences of the facts and the rules at each time period in succession. Continuity is established by the fact that many of these rules will refer to facts established in previous time periods.

The logic of SDML's propositional inference machinery is equivalent to a fragment of Konlige's strongly grounded autoepsitemic logic (SG-AE) [5]. This is the logic of reasoning with introspection - the ability to reason about one's own beliefs. Belief for a human reasoner corresponds to *inference* in SDML. This mainly effects the treatment of negation, which in SDML is implemented with the notInferred primitive.

In most normal circumstances this acts as normal negation, since all positive logical implications of the model are inferred, if something is not true (from the point of view of the model) then it will be not inferred. In some circumstances, you are allowed to start with an *assumption* that something is not inferred until shown otherwise (usually by another rule firing). If rules conflict with each other, SDML can re-trace its inference and try another possible SG-AE extension of the original model. If no such extensions are possible then an error message arises and the simulation halts.

Thus, in the absence of logical errors by the modeller, SDML produces a possible animation of the modeller's original rules and facts given some reasonable assumptions about the model.

8 Logical Language and Informal Description

Logical language is much closer to a natural language description of agent behaviour than, say, Bayesian probability theory. Thus the declarative nature of SDML enables the specification of agent behaviour with the minimum of distortion. Similarly the results of the simulation are stored in the form of declared 'facts' on the various databases, in the language of SDML and the clauses specifically designed by the modeller. The results are much easier to interpret, especially when they involve qualitative information. We do not

3 Except for those that are written by meta-level rules, which are themselves specified by the modeller.

claim that such interpretations are impossible with imperative and numerically based languages, but that they are more cumbersome to use in this way compared to the more natural manner afforded by a declarative framework and programming language.

Like any formal system, a large part of the benefit of the modelling process is that requires the underlying assumptions to be made explicit. Since the process of translation from an informal description or intuition to the language of SDML and back again in the interpretation of results is much easier and more natural in such a language, it is much more likely that both assumptions and the interpretation of the results will not be hidden by the formal machinery. We have found this approach to provide the basis for a more productive dialogue between modeller and model than is possible with imperative models and languages.

Similarly the results of an SDML model are transparently revealed in terms of true statements aiding interpretation of the results.

Such a style of modelling makes it easier to prove theorems about the limitations of model behaviour in each case — a property which is important both for theoretical justification and practical understanding of the processes we are modelling (cf. Axtell and Epstein,1994). While SDML and the modelling process give insights into what is possible, providing the basis for new languages of discourse about the processes being investigated, theorem proving can enable statements to be made about what is necessarily true about of model.

We believe that this process of theorem proving can be substantially automated. Once the modeller has gained an understanding into the nature of the models, hypotheses can be made about that model and the automatic theorem prover can be used to attempt to prove them in the logic. This can also have the useful effect of revealing further assumptions necessary to prove the hypotheses[4].

9 A Modelling Methodology using SDML

The formal tools of logical modelling and the computer tool of SDML allow the modeller to go beyond the limitations of statistical or differential equations and begin to tackle the complexity inherent in many of the management and business situations faced today.

---

[4]An example of such benefits arising from use a theorem prover can be found in [6].

A possible outline for the use of such tools might be the following.

1. Choose a target for modelling for which a formal model of its separate causal parts is credible.
2. The intuitions of the causes of the agents' actions are encoded into the rules governing their "actions" in the formal system (carefully remembering any assumptions).
3. These rules are then animated by the deductive machinery of SDML.
4. The results are compared to the known data on the effects of this process, according to previously decided criteria.
5. If the results do not match the data according to these criteria, then go back to step (2). Ensure that if a loop of steps (2)-(5) occurs too many times then you more fundamentally change the model (or give up on this process).
6. If the results are successful examine the emergent process of the animation by the formal system, interpreting it into qualitative conclusions (minding your assumptions).
7. If appropriate make hypotheses about the model and attempt to prove them.
8. Recursively apply any insights to further modelling of this kind, to more traditional types of models or other informal decision making.

This process is illustrated in figure 3 below.

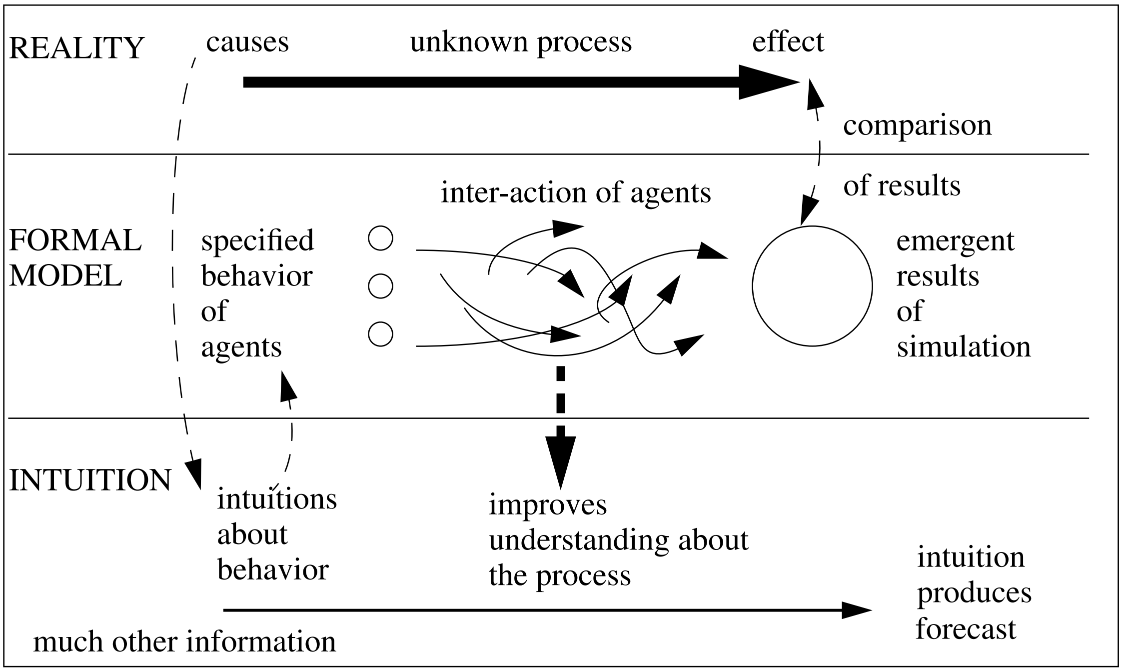

Figure 3: A Methodology for simulation

Thus we can continue to make substantial progress in domains where traditional analytic modelling tools are impractical.

10 Conclusion

In many modelling situations analytic solutions or traditional "black-box" solutions are unavailable. The same situations are often those where the understanding of the processes involved are most lacking. The combination of the qualitative logical formal tools combined with a suitable declarative modelling language allows for a coherent methodology to be applied which will allow a new understanding of these processes to evolve, and new results to be obtained.

References

[1] Axtell, R.L. and Epstein, J.M., (1994), "Agent-based Modelling: Understanding our Creations", *The Bulletin of the Santa Fe Institute*, 9, 28-32.

[2] Cohen, P.R. (1985), *Heuristic Reasoning Under Uncertainty*, London: Pitman.

[3] Edmonds, B. and Moss, S, (1996). The Credible Modelling of Economic Agents with Limited Rationality, AIEM96, TEl Aviv, 1996.

[4] Fox, J., D. Clark, A. Glowinski and M. O'Neil (1990), "Using Predicate Logic to Integrate Qualitative Reasoning and Classical Decision Theory", *IEEE Transactions on Systems, Man and Cybernetics* **20** (2), pp. 347-357.

[5] Konolige,K (1992): Autoepistemic Logic. In: Handbook of Logic in Artificial Intellegence and Logic Programming. Vol. 3. (Eds: Gabbay,DM; Hogger,CJ; Robinson,JA) Clarendon Press, Oxford, 217-295.

[6] Masuch, M. and Huang, Z, (1994), "A Logical Deconstruction of Organizational Action: Formalizing J.D. Thompson's *Organisations in Action* in a Multi-Agent Action Logic", CCSOM Working Paper 94-120, Department of Statistics and Methodology, University of Amsterdam.

[7] Wallis, S. and Moss, S. (1995), "Efficient Forward Chaining for Declarative Rules in a Multi-Agent Modelling Language", Centre for Policy Modelling Discussion paper CPM-4. Available electronically at http://www.fmb.mmu.ac.uk/cpm/cpmrep04.html.