AGENT DESIGN METHODS

Michael Wooldridge   Simon Parsons

Department of Computer Science
University of Liverpool, UK

Overview

- How do we:
  - *specify*
  - *implement*; and
  - *verify*

  agent based systems.
- Especially in the context of constructing information ecosystems. . .
- . . . which contain thousands of agent-inhabitants.

Specifying Agent Systems

- Agent specifications are generally given in a quantified multi-modal logic, containing some or all of the following features:
  - modalities for information attitudes;
  - modalities for pro-attitudes;
  - representation of action;
  - representation of time.
- *The key problem in developing a framework for specifying agent theories is to sort out the relationships between these different aspects of agency.*

  For example, even the relationship between knowledge and belief is the subject of debate! The relationship between action and other attitudes is *very* troublesome.

Implementing Agent-Based Systems

- *Given a specification φ, expressed in some logical language L, how do we construct a system S such that $\{S\} \models \varphi$?*
- (Note that we assume a *logical* agent specification.)
- There seem to be 3 possibilities:
  1. manually refine φ to executable format;
  2. directly execute φ;
  3. compile φ into a directly executable form.

Manual Refinement

- To all intents and purposes, manual refinement *is* software engineering!
- The predominant approach to SE today.
- But how to refine intentional specification to code?
- No obvious answers—all existing implementations require a "leap of faith" from specification to implementation.

Executing Agent Specifications

- What does it mean, to execute a formula φ of logic L?
- It means *generating a logical model, M, for φ, such that $M \models_L \varphi$*.
- If this generation is done without interference — if the agent has complete control over its environment — then execution reduces to *constructive theorem proving*:

  show that φ is satisfiable by building a model for φ.

- However, agents are not interference-free; they must construct a model in the presence of input from the environment.

- Model-building is thus an iterative process:
  - environment makes something true;
  - agent responds by making something else true, in such a way as to satisfy φ;
  - environment responds; ...
- Execution and theorem proving are closely related

$$\Rightarrow$$

execution of sufficiently rich (quantified) languages not possible.

- At first sight, it might seem that suggesting the direct execution of complex agent specification languages is naive; that it is exactly the kind of suggestion that detractors of symbolic AI hate.
- One should be very careful about what claims or proposals we make; in particular, there has been no work done on directly executing the kind of multi-modal logic that we use to specify agents.
- However, in certain circumstances, execution of interesting agent specification languages *is* possible. (eg Concurrent Metatem).

Compiling Agent Specifications

- An alternative to direct execution is *compilation*.
- In this paradigm, we take a specification φ, and do a constructive proof of the satisfiability of φ — this involves generating a logical model for φ.
- Models for modal logics are typically directed graphs, which closely resemble automata.

  Thus, the idea goes, generate a model for φ, and read off the automata for φ from the model that results.

  You end up (hopefully) with a very simple machine that does not do complex symbolic reasoning.

- Much similar work in mainstream computer science:
  - [Manna and Wolper, 1984] generate *synchronization skeletons* for concurrent systems from linear TL specifications;
  - [Pnueli and Rosner, 1989] generate reactive modules from branching TL (CTL*) formulae;

  Conclusions of this work: it's hard!

  Works well in the propositional case — not well at all in general.

  Again, we run up against undecidability of quantified logic, and the general difficulty of theorem proving.

Verifying Agent-Based Systems

- Finally, given a system S and specification φ, how do we *verify* S with respect to φ, i.e., how do we show that $\{S\} \models \varphi$?
- The semantic approach to this is via model checking.
- In mainstream computer science, model checking means:
  - taking a program Π, and from that program generating the model M_Π corresponding to that program;
  - showing that $M_\Pi \models \varphi$, i.e., that the specification formula φ holds in M_Π, and thus that the program satisfies its specification.
- Model checking is generally reckoned to be easier than theorem proving!

The SLIE Approach

- Work on SLIE fits into this overall picture in a number of ways.
- In fact, it fits in all three of the main areas identified above
  - in specifying agent systems;
  - in implementing agent systems; and
  - in verifying agent systems.
- The work both extends existing approaches, and provides new ways to deal with current problems.

SLIE and Specification

- SLIE has invesitigated a number of approaches for specifying agent systems.
- The aim is to get a feel for how different languages work in the specification of ecosystems.
- SLIE has also developed new languages for specification:
  - ISLANDER
  - MABLE
- Currently these are seen as complementary:
  - suitable for different things; and
  - able to be used in combination.

SLIE and Implementation

- Given a specification we have to:
  - manually refine it;
  - directly execute it; or
  - compile it into an executable form.
- Each of these aspects has been addressed in SLIE.
- Both the equation-based modelling and the use of institutions provide support for the manual refinement of high-level specifications.

- The equation-based approach makes it possible to run simulations from a high-level description of the system.
- This allows the parameters of the system itself to be refined before any detailed design is carried out.
- Institutions give an intermediate level of refinement between the overall system and the individual agents.
- Designing an institution involves designing the interactions between agents, focussing attention at the level of groups of agents.
- Once complete, the institution design provides constraints on the design of the agents themselves.
- Thus the output of the institution design is a set of “agent templates” for agents which can interact within a given set of institutions.

- MABLE permits a limited form of direct execution.
- We are working towards the compilation of MABLE and ISLANDER specifications into executable form.
- These executable forms will then be run on the experimental platform developed in SLIE.
- We are also seeking to be able to compile MABLE programs into a form where they can be directly executed on any platform.
- Here we can view MABLE as an agent programming language rather than a specification language.
- ISLANDER institution specifcations could then be compiled into MABLE programs, fleshed out, and then executed.

SLIE and Verification

- Agent specifications in MABLE can be compiled into the language PROMELA which can then be model-checked using the SPIN system.
- This provides a way of ensuring that agents have the behaviours that are required, and only the behaviours that are required.
- Compiling ISLANDER specifications into MABLE will allow the interactions established at the institution level to be verified in the same way.
- Verification is also possible between the results of the equation-based modelling and the results of compiling and executing MABLE/ISLANDER specifications.

References

[Harel, 1984] Harel, D. (1984). Dynamic logic. In Gabbay, D. and Guenther, F., editors, *Handbook of Philosophical Logic Volume II — Extensions of Classical Logic*, pages 497–604. D. Reidel Publishing Company: Dordrecht, The Netherlands. (Synthese library Volume 164).

[Manna and Wolper, 1984] Manna, Z. and Wolper, P. (1984). Synthesis of communicating processes from temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 6(1):68–93.

[Pnueli and Rosner, 1989] Pnueli, A. and Rosner, R. (1989). On the synthesis of an asynchronous reactive module. In *Proceedings of the Sixteenth International Colloquium on Automata, Languages, and Programs*.

[Rao and Georgeff, 1993] Rao, A. S. and Georgeff, M. P. (1993). A model-theoretic approach to the verification of situated reasoning systems. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI-93)*, pages 318–324, Chambéry, France.