Developing and Evaluating Cognitive Architectures with Behavioural Tests

Peter C. R. Lane
School of Computer Science, University of Hertfordshire,
College Lane, Hatfield AL10 9AB, Hertfordshire, UK
Peter.Lane@bcs.org.uk

Fernand Gobet
School of Social Sciences, Brunel University,
Uxbridge UB8 3PH, Middlesex, UK
Fernand.Gobet@brunel.ac.uk

Abstract

We present a methodology for developing and evaluating cognitive architectures based on behavioural tests and suitable optimisation algorithms. Behavioural tests are used to clarify those aspects of an architecture's implementation which are critical to that theory. By fitting the performance of the architecture to observed behaviour, values for the architecture's parameters can be automatically obtained, and information can be derived about how components of the architecture relate to performance. Finally, with an appropriate optimisation algorithm, different cognitive architectures can be evaluated, and their performances compared on multiple tasks.

Introduction

Cognitive architectures are an important component of several disciplines, including: artificial intelligence, cognitive science and psychology. Within these disciplines, architectures are variously used as: an explanatory framework, from which to deduce reasons for observed behaviours; an implemented 'agent,' which can be observed to carry out some task; and, a system for making predictions, which can be used to test the principles of the architecture. Our interest here is in how a cognitive architecture can be *evaluated*. In particular, we focus on the use of cognitive architectures to generate working models (or agents), which are then given some task to perform. The architecture's quality is judged based on whether the models it generates capture some aspect of intelligent behaviour.

One specific motivation for developing cognitive architectures is to create a *unified theory of cognition* (Newell 1990) – a cognitive architecture which explains a wide range of intelligent behaviours, in a domain-independent fashion. We shall focus on two issues regarding the scientific acceptability of cognitive architectures: their implementation, and their parameters. We propose a methodology for improving the quality of an implementation, and show how this supports automated techniques for analysing the architecture's parameters. Finally, we show how our automated techniques enable empirical comparisons to be made between different cognitive architectures.

The key elements of our methodology are:

1. develop the implementation of a cognitive architecture using a *robust-testing methodology* (Lane & Gobet 2003);
2. optimise the parameters of the architecture, using a suitable search technique (Gobet & Lane 2005; Lane & Gobet 2005a; 2005c);
3. use knowledge-discovery techniques to determine the important variables (Lane & Gobet 2005b); and
4. use data-analysis and visualisation techniques to compare the different architectures.

We explain these four elements in the following sections, after briefly introducing a case study in developing a model of categorisation, which we use to illustrate the ideas and show their benefits. We finish by relating our methodology to open questions on the evaluation of cognitive architectures.

Case-Study: Models of Categorisation

As a case study, we have used the problem of developing a model of categorisation. Our models are drawn from four different theories: two mathematical theories, neural networks (McLeod, Plunkett, & Rolls 1998) and CHREST (Gobet *et al.* 2001). Categorisation is the problem of assigning instances into categories. We use the five-four task, and specifically the experimental data collected by Smith and Minda (2000) from 29 different experiments on humans. A theory is judged on its ability to produce models which match the performance of human participants in a wide range of experiments and measures, such as response time and number of errors. Each experimental result forms a constraint on the range of admissable models provided by a theory. We define an experimental *constraint* from the following information:

1. The experimental setting, consisting of its stimuli, and the separation of the stimuli into training and transfer sets.
2. The data collected from the participants.
3. The measure of fit used to compare the model's performance with the participants'.

Each constraint can then be applied to a model by running the model on the experiment, collecting similar data to that obtained from the participants, and then computing the measure of fit between the performances of the model and that of the participants. In Figure 1 and Table 1 constraints are referred to by 'SSE 1st', 'AAD Time' etc; SSE/AAD refer to the computation of fit (Sum-Squared Error/Average Absolute Deviation), and 1st/Time to the data compared against.

Robust-Testing Methodology

Cognitive architectures are typically implemented as computer programs, which are then run and their behaviour observed. As running architectures are computer programs, we can look for lessons in how software engineers ascertain the correctness of their programs. There are two basic approaches, *prescriptive* and *descriptive*.

Prescriptive approaches rely on a high-level specification language to formalise what the program is meant to achieve. For example, specifications in the Z language (Lightfoot 2001) can be used to define cognitive architectures, as was done with Soar (Milnes 1992). The problem with such definitions is that they require a new implementor of a cognitive architecture to *prove* that their implementation meets the specification. Proofs of program behaviour are difficult, and may themselves be flawed.

Descriptive approaches are better because they automate the checking process in a set of tests. Test-driven development requires that a program is developed along with a set of tests, which verify the program's behaviour. In Lane and Gobet (2003) we highlighted two groups of tests important for verifying a cognitive architecture: these behavioural tests are *process tests* and *canonical results*. The process tests verify that the implementation correctly carries out the processes described in the cognitive architecture, and the canonical results verify that the complete implementation has the desired, observed behaviour. For example, with a neural network, the process tests may confirm that a particular weight update is carried out correctly, whereas the canonical results will verify that the network produces a pattern of behaviour similar to that observed with humans.

The behavioural tests are "designed to satisfy the user that the [program] does indeed implement the intended cognitive theory" (Lane & Gobet 2003). Developing an implementation with explicit tests is not simply a matter of good programming practice, but enhances the scientific value of the implemented architecture. Firstly, the tests explicitly document the desired behaviour of the model. This enables a programmer to identify elements of the implementation which are critical to the theory, and those which are not. Secondly, the tests support a user trying to *reproduce* an implementation. In many sciences, all aspects of an experiment should be reproduced before results can be accepted. Supporting a user in creating a new implementation will aid in clarifying the nature of the architecture and, most importantly, ensure that hidden details of the implementation do not affect the interpretation of results; some difficulties in reproducing the implementation of the Soar cognitive architecture are reported by Cooper and Shallice (1995).

The third and final way in which behavioural tests enhance the scientific value of the implemented architecture is that they support natural algorithms to optimise, explore and compare the parameters in this and other architectures. Our proposed methodology further uses tests to improve the empirical analysis of the behaviour of cognitive architectures on multiple tasks. The experimental constraints, described above, form the set of 'canonical results' for our models to meet, and are not architecture specific; hence, canonical results can be used to compare different architectures.

Optimising Parameters

The ultimate goal of developing cognitive models is to produce better theories of human behaviour. However, most theories in psychology and cognitive science have a number of parameters, and this number can sometimes be considerable. In spite of some efforts in this direction (Ritter 1991; Tor & Ritter 2004), few modellers routinely use formal, automated techniques to optimise the parameters and/or the structural components of their models. It is unclear whether this state of affairs is due to researchers' unawareness of optimisation techniques regularly used in engineering and in other scientific fields, or whether it is due to the fact that these techniques have not been tailored to the specific challenges of developing cognitive models, such as the complexity of the models used, the number of parameters present, the noise in the empirical data, and so on. However, the fact that most cognitive models are not optimised poses important problems for building and evaluating theories. For example, what is the meaning of comparing two theories if their parameters have not been optimised (Gobet & Ritter 2000; Ritter 1991)? One of the goals of our research is to develop a method of optimising cognitive architectures, tailored to the needs of cognitive scientists.

The problem of locating an optimal cognitive model is usefully considered as an *optimisation problem*. In addition, because we require each model to perform as well as possible in multiple experiments, the problem is a *multi-objective* optimisation problem. However, in such problems it is not always possible to rank two different solutions: one may be better on one objective, but the other better on a second. Hence, we look for a set of solutions, where no member of the set can be outperformed everywhere by another solution; this set is known as the *pareto-optimal set*, and will contain the optimal models. In common with other applications, the nature of our models precludes the use of an analytic technique to calculate the pareto-optimal set. Instead, search techniques, which gradually approximate the pareto-optimal set, are used to find approximations to the optimal models.

There exist a range of techniques for finding pareto-optimal sets in the literature. Genetic algorithms (Holland 1975) are one of the must suitable techniques for multi-objective optimisation, due to the parallel nature of their search, and several different such algorithms are used; e.g. see Coello Coello (2000; 2003) for an overview.

Locating Predictive Parameters

Some parameters will have *optimal* values, especially in relation to particular tasks. We can find these associated parameter-task pairs by analysing the collection of models created during optimisation. For example, Figure 1 shows the performance of the time parameter for 10,000 instances of the neural-network type of model evaluated on two different tasks. As is readily apparent, the performance of the model has a clear global minimum for task 'AAD Time' (the lower graph), but no such optimum value is apparent for task 'SSE 1st' (the upper graph). A statistical measure, Pearson's product moment correlation, can locate those param-

Architecture	Parameter	Value	Task	Correlation
Context	response time	1174.74	SSE Time	0.97
Prototype	response time	1129.73	AAD Time	0.99
Connectionist	response time	1130.62	AAD Time	0.98
CHREST	reaction time	242.52	AAD Time	0.82
CHREST	sorting time	512.60	AAD Time	0.86

Table 1: (Selected) reported parameter values and tasks with high (> 0.8) correlation.

eters which take on optimal values in individual tasks. By storing every model tested, along with its fitness on all of the tasks, we test the degree to which any parameter's value correlates with task performance. Specifically, we find the value of the parameter, p, in the stored models which corresponds to the lowest fitness value for each test. The degree of correlation is then computed between the value of the fitness and the absolute difference of each parameter value from p. A high degree of correlation (> 0.8) means that the parameter acts like the lower graph in Figure 1.

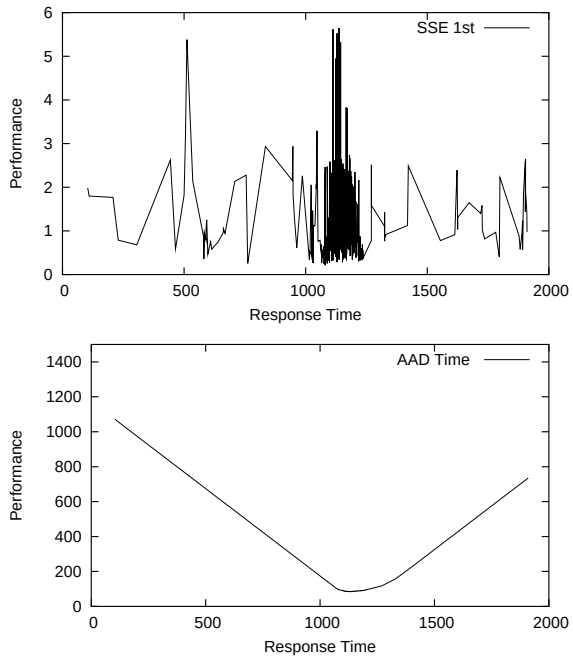

Figure 1: Graph of performance of 10,000 neural-network models against parameter value on two categorisation tasks.

The correlations detected in this manner are useful in explaining what elements of the architectures are most important in particular tasks. Table 1 lists selected correlations detected by the system. The correlations mostly agree with what might be expected. There are strong, and readily apparent, correlations between the different models' timing parameters and the task involving a timed response. No behavioural effects were apparent for most of the other parameters (Lane & Gobet 2005b).

Comparing Architectures

A unique feature of our case study and the scientific goal of comparing cognitive architectures, is that different kinds of model will be compared. For example, neural-network models will be compared with CHREST models. We have discussed the challenges in managing a population of different theories elsewhere (Lane & Gobet 2005c). More recently, we have resolved the challenges by maintaining separate, equal-sized populations for each type of theory, or cognitive architecture, and computing comparisons (for calculating the non-dominated members of the pareto-optimal set) across all the populations. This 'Speciated algorithm' produces optimal members from all competing theories.

Discussion

Our robust-testing methodology provides an initial answer to the question 'How should we validate the design of a cognitive architecture?' A key element in validating an architecture is its implementation. We argue that it is not just 'good practice' to provide tests for software, but also important scientifically. The tests provide a means to document the key behaviour of the architecture within the implementation, making it easier to understand, and easier to reproduce (Lane & Gobet 2003). An additional benefit is that two kinds of behavioural tests support optimisation of parameters and comparison between architectures.

By making important behaviours explicit in experimental constraints (as described above), we also provide an answer to the question 'How are architectures to be compared in an informative manner?' Before we can begin qualitative evaluation of two architectures, it must first be clear that they both achieve the desired empirical results. Our optimisation algorithms also ensure that examples from different architectures are suitably optimised, and multiple examples of each are provided for detailed comparison. As an aside, there is clear evidence that using automated optimisation techniques produces better results than hand optimisation: we routinely find a 40% improvement in our case study over figures reported in the literature, supporting the earlier findings of Ritter (1991).

Comparing multiple architectures simultaneously enables us to analyse their performance, and so consider the questions: 'How can we determine what architectures to use for different tasks or environments? Are there any trade-offs involved?' Even in our small case study, we found that different theories performed better on some tasks than on others.

More interestingly, our methodology enables us to identify properties of an architecture which are suited for differ-

ent tasks. For example, in our case study we used a technique to seek a correlation between one parameter's value and behaviour. This produced some natural correlations, between parameters relating to time and tasks measuring time taken. Something to be improved is to seek correlations between groups of parameters and behaviour. Most parameters, such as the weights within the mathematical models, work together to produce a behaviour, and more sophisticated analysis techniques are required to locate such dependencies; individually, their correlation with task performance was of the order of 0.3.

However, it may also be the case that for many of the parameters, e.g. the learning rate for a neural network, there simply is no correlation of the parameter value with the model's performance. This would make the parameter a 'free' variable; one needed to make the implementation work but which has no explanatory power. Identifying critical and free variables is important in understanding how an architecture relates to its environment and other architectures.

Conclusion

We have argued that a correct, scientific use of cognitive architectures requires careful evaluation of the *implementation* of that architecture, automated *optimisation* of its parameters, and semi-automated *comparison* of the performance of different architectures. Some of these ideas have been implemented and tested in a system to evolve models from four theories of categorisation, fitting the models against psychological data; this system can be downloaded from: `http://homepages.feis.herts.ac.uk/~comqpcl/evolving-models.html`.

In conclusion, we propose the following guidelines for working with cognitive architectures, particularly when modelling observed behavioural data:

1. Use the robust-testing methodology to document important elements of the implementation.
2. Use optimisation to ensure parameters are set to their best values.
3. Use data-analysis and visualisation techniques on the range of models generated during optimisation, to better evaluate aspects of the architectures and their relation to particular behaviours.
4. Compare different types of architectures, to locate those behaviours which are common to multiple theories, or specific to only one.

References

Coello Coello, C. A. 2000. An updated survey of GA-based multiobjective optimization techniques. *ACM Computing Surveys* 32:109–143.

Coello Coello, C. A. 2003. Recent trends in evolutionary multiobjective optimization. In Abraham, A.; Jain, L.; and Goldberg, R., eds., *Evolutionary Multi-Objective Optimization*. London, UK: Springer-Verlag. 7–32.

Cooper, R., and Shallice, T. 1995. Soar and the case for unified theories of cognition. *Cognition* 55:115–49.

Gobet, F., and Lane, P. C. R. 2005. A distributed framework for semi-automatically developing architectures of brain and mind. In *Proceedings of the First International Conference on e-Social Science*.

Gobet, F., and Ritter, F. E. 2000. Individual data analysis and Unified Theories of Cognition: A methodological proposal. In Taatgen, N., and Aasman, J., eds., *Proceedings of the Third International Conference on Cognitive Modelling*, 150–57. Veenendaal, The Netherlands: Universal Press.

Gobet, F.; Lane, P. C. R.; Croker, S. J.; Cheng, P. C.-H.; Jones, G.; Oliver, I.; and Pine, J. M. 2001. Chunking mechanisms in human learning. *Trends in Cognitive Sciences* 5:236–243.

Holland, J. H. 1975. *Adaptation in natural and artificial systems*. Ann Arbor: The University of Michigan Press.

Lane, P. C. R., and Gobet, F. 2003. Developing reproducible and comprehensible computational models. *Artificial Intelligence* 144:251–63.

Lane, P. C. R., and Gobet, F. 2005a. Applying multi-criteria optimisation to develop cognitive models. In *Proceedings of the UK Computational Intelligence Conference*.

Lane, P. C. R., and Gobet, F. 2005b. Discovering predictive variables when evolving cognitive models. In Singh, S.; Singh, M.; Apte, C.; and Perner, P., eds., *Proceedings of the Third International Conference on Advances in Pattern Recognition, part I*, 108–117. Berlin: Springer-Verlag.

Lane, P. C. R., and Gobet, F. 2005c. Multi-task learning and transfer: The effect of algorithm representation. In Giraud-Carrier, C.; Vilalta, R.; and Brazdil, P., eds., *Proceedings of the ICML-2005 Workshop on Meta-Learning*.

Lightfoot, D. 2001. *Formal Specification Using Z*. Basingstoke, UK: Palgrave.

McLeod, P.; Plunkett, K.; and Rolls, E. T. 1998. *Introduction to Connectionist Modelling of Cognitive Processes*. Oxford, UK: Oxford University Press.

Milnes, B. 1992. The specification of the Soar cognitive architecture using Z. Technical report: CMU-CS-92-169, Carnegie-Mellon University.

Newell, A. 1990. *Unified Theories of Cognition*. Cambridge, MA: Harvard University Press.

Ritter, F. E. 1991. Towards fair comparisons of connectionist algorithms through automatically optimized parameter sets. In Hammond, K. J., and Gentner, D., eds., *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society*, 877–881. Hillsdale, NJ: Lawrence Erlbaum.

Smith, J. D., and Minda, J. P. 2000. Thirty categorization results in search of a model. *Journal of Experimental Psychology* 26:3–27.

Tor, K., and Ritter, F. E. 2004. Using a genetic algorithm to optimize the fit of cognitive models. In *Proceedings of the Sixth International Conference on Cognitive Modeling*, 308–313. Mahwah, NJ: Lawrence Erlbaum.