Appeared as: Ritter, F. E., & Wallach, D. P. (1998). Models of two-person games in ACT-R and Soar. *In Proceedings of the Second European Conference on Cognitive Modelling.* 202-203. Nottingham: Nottingham University Press.

Models of Two-person Games in ACT-R and SOAR

Frank E. Ritter
University of Nottingham
Nottingham NG7 2RD, UK
frank.ritter@nottingham.ac.uk

Dieter P. Wallach
Saarland University
66041 Saarbrücken, Germany
dwallach@cops.uni-sb.de

We were interested in understanding and comparing how ACT-R (Anderson & Lebière, in prep.) and SOAR (Newell, 1990) could each model a given dataset. We analyze and compare two models in their ability to account for a classical 2 person game, including the effort necessary to create and run them. In comparing the models and their results we provide two sample models and start to explore the potential role of abstract models and different types of data.

Game description. In two player, 2x2 games each player can choose one of two alternatives in each round. The players are rewarded according to a payoff matrix. The prisoner's dilemma is an example of such a 2 person game.

We used data from a classical experiment (Suppes & Atkinson, 1960) of how people learn when they play a normal form, two player 2x2 game with a nontrivial unique mixed strategy equilibrium. Table 1 shows the payoff matrix used in the experiment that we model here. This matrix has a unique mixed strategy equilibrium point, that is, a stable set of strategies, when Player 1 chooses option A1 with probability 1/3 and player 2 chooses option A2 with probability 5/6. Figure 1 shows the empirical choice frequencies of option A for player 1 (A1) and player 2 (A2) aggregated in 5 blocks with 40 rounds each, of 20 pairs of participants playing the game for 200 rounds (Erev & Roth, 1998).

ACT-R model. Figure 2 shows the structure of the ACT-R model used to account for this data. For a full description of the ACT-R model see Bracht, Wallach and Lebière (1998). The model consists of two simple productions for each player representing the options available:

Rule1: If Player 1 chooses => choose Option A.
Rule2: If Player 1 chooses => choose Option B.

In every round, both of these productions are applicable for each player modeled. ACT-R's subsymbolic cost learning mechanism learns the relative payoff of each production rule and updates their expected gain based on the outcome of the round. In general, ACT-R selects the production rule with the highest expected gain. Two architectural parameters were used to fit the model to the data (*expected gain noise* and *number of previous production applications*). The model with the same parameter settings has also been applied successfully to data from three other experiments taken from Erev and Roth (1998).

		Player 2 Option A	Option B
Player 1	**Option A**	2, 4	6, 0
	Option B	3, 3	1, 5

Table 1. Payoff matrix used by the models here.

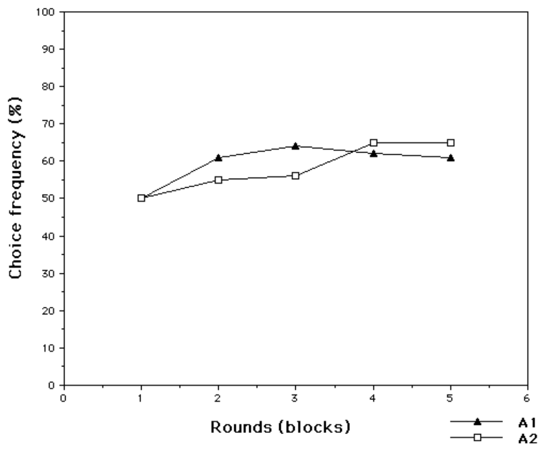

Figure 1. The evolution of strategies in the subjects on the Table 1 payoff matrix

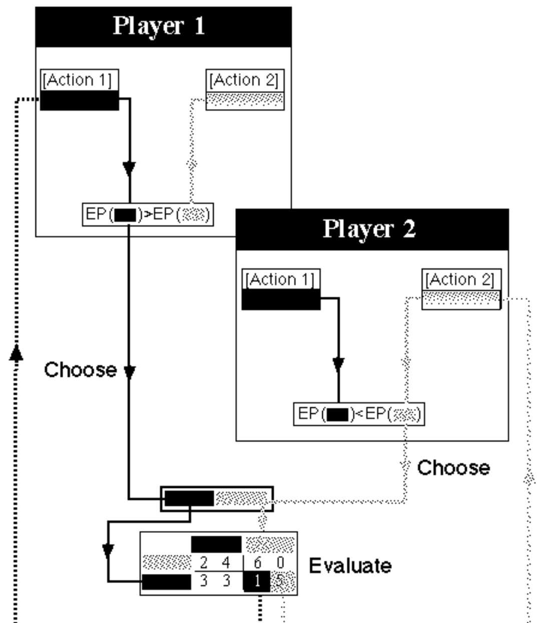

Figure 2. Description of the ACT-R model.

SOAR model. The easiest way to explore a SOAR model of this task is to create an abstract model. An abstract model is based on an information processing model or architecture. It predicts what a running model would do, without implementing the internal behaviors (e.g. Langley, 1996; Ohlsson & Jewett, 1994).

An abstract model of the simplest SOAR model could start with a single operator representing each choice. Each round, an operator is randomly chosen to apply. After each round, the expected values of each of the four payoffs occurring can be computed for each player. Operators that do better than the average payoff can be duplicated through a reflection-like process (not specified, but similar to the process in Bass et al., 1995). Various other ways of duplicating operators are possible (e.g. duplicate operators as many times as their payoff). In SOAR these processes are determined not by the architecture but by knowledge. It is fairly straightforward to implemented a program to compute the expected population of operators on each round. The results of this program are shown in Figure 3. While this model is not currently based on a running Soar model, creating such a model should be straightforward. Deriving its predictions is much simpler as an abstract model, for programming an interface to record multiple rounds and games would be less straightforward.

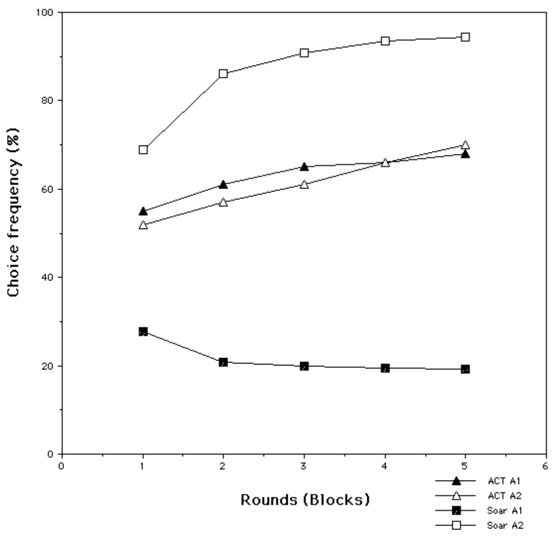

Figure 3. The evolution of strategies in the two models on the Table 1 matrix.

Comparisons

Model fit. As Figure 1 shows, the ACT-R model captures the general tendencies in the empirical data quite nicely. In addition to this *short term* prediction, the model converges asymptotically to the equilibrium of classical game theory in the *long term* (after >1500 rounds). The initial Soar model, on the other hand, does not match the subject data (short term) nearly as well, but instead appears to quickly converge to near the equilibrium.

Effort. Both models took about the same time to implement (4-5 hours), including the ability to automatically run and trace the model. Both models can run 200 rounds of 20 subject pairs in under 30s.

Abstract models. The Soar model would not be as easy to run if it was implemented in Soar productions. It would not be straightforward to implement an abstract version of the ACT-R model based on its current mechanism, but it is easy to create an abstract model of the operator population model in ACT-R (as a rule population), or an ACT-R model directly based on this principle. The difficulty of creating abstract models within each architecture varies by task, but appears to be generally easier in SOAR. Creating full models appears, however, to be more difficult. In this task, the SOAR architecture appears to have less to say than ACT-R because it lacks architectural mechanisms to account for the learning observed here. While the Soar model does not match nearly as well (yet), it allows the space of possible models to be explored quite quickly (about 5 min. per model).

Conclusions

These results are very interesting, for they start to suggest possible trade-offs in modeling; between abstract and information processing models, and between architectures. This work also emphasizes the role of usability as a necessary precondition for explorations of this kind.

REFERENCES

Anderson, J. R. & Lebière, C. (in prep.). *The atomic components of thought.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Bass, E. J., Baxter, G. D., & Ritter, F. E. (1995). Using cognitive models to control simulations of complex systems. *AISB Quarterly*, 93, 18-25.

Bracht, J., Wallach, D. & Lebière, C. (1998). On the need and performance of cognitive game theory: ACT-R in experimental games with unique mixed strategy equilibria. To appear in *The Economic Science Association (ESA) Annual Conference.*

Erev, I. & Roth, A. (1998, in press). Predicting how people play games: Reinforcement learning in experimental games with unique, mixed strategy equilibria. *American Economic Review.*

Langley, P. (1996). An abstract computational model of learning selective sensing skills. In *Proceedings of the 18th Annual Conference of the Cognitive Science Society.* 385-390. Hillsdale, NJ: Lawrence Earlbaum Associates.

Newell, A. (1990). *Unified theories of cognition.* Cambridge, MA: Harvard University Press.

Ohlsson, S. (1995). Abstract computer models: Towards a new method for theorizing about adaptive agents. In *Machine Learning: ECML-95.* Berlin: Springer-Verlag.

Suppes, P. & Atkinson, R. C. (1960). *Markov learning models for multiperson interactions.* Stanford, CA: Stanford University Press.