How to Construct a Believable Opponent using Cognitive Modeling in the Game of Set

Niels A. Taatgen (niels@ai.rug.nl)
Marcia van Oploo (marcia@ai.rug.nl)
Jos Braaksma (j.braaksma@ai.rug.nl)
Jelle Niemantsverdriet (jelle@niemantsverdriet.nl)
Department of Artificial Intelligence, Grote Kruisstraat 2/1
9712 TS Groningen, Netherlands

Abstract

An interesting domain of application for Cognitive Modeling is the construction of computer opponents in games. We present a model of the game of Set. The model is sensitive to the difficulty of the situation in the game, and can explain the difference between beginners and experts. Furthermore, the model is used in a computer game in which players can play Set against various versions of the model.

Introduction

In computer games, the human player often faces one or more computer opponents. In order to make the game enjoyable, these computer opponents have to be intelligent, otherwise they wouldn't be much of a challenge. The classical example of playing a game against the computer is chess, and the focus of designing a computer chess player has always been to have a player that plays as good as possible. In fact, since Deep Blue beat Kasparov the interest in computer chess seems to have diminished, but maybe the recent rematch against Kramnik will change matters. Nevertheless, human chess players complain that computer chess programs are no fun to play against. A possible reason for this is that computer chess programs have long left the approach of mimicking human chess players, but instead focused on brute-force techniques.

This leads us to the idea that a computer opponent becomes more interesting and enjoyable to play against as it behaves more like a person. In this paper we will explore this idea and demonstrate how cognitive modeling can help to produce more interesting computer opponents. The basic idea is relatively simple: study how people (preferably at different levels of skill) behave in the game you want a computer opponent of, make a cognitive model of this behavior that closely matches human behavior, and incorporate this in a computer program.

The game we will use is the game of Set. Set is a card game that is quite trivial to play perfect for a computer program, so the challenge is to model an opponent that is interesting to play against. Another interesting aspect of Set is that it combines several aspects of cognition: perception, information processing, strategy, learning and time pressure. We will describe an experiment and several possible models of playing Set, and incorporate those in a program that can one can play against.

The game of Set

Set[1] is played with a deck of 81 cards, and with as many players as can sit around the table. Cards in Set have pictures made out of symbols on them that can be described by four attributes: color, shape, filling and number. Each of the four attributes can have three possible values, blue, green and red for colors, rectangle, oval or wiggle for shape, solid, open or speckled for filling, and one, two or three for number. So two solid red ovals is a card, as is three blue speckled wiggles. Of each possible combination of attributes and values there is one copy in the deck, hence 3 x 3 x 3 x 3 = 81 cards. In the game, twelve cards are dealt open on the table, and the goal for the players is to be the first to find a set, a combination of three cards that satisfies the following rule:

For each attribute, the three values that the cards have on this attribute are either all the same, or all different.

An example of a set is: one solid red oval, two solid red ovals and three solid red ovals. Another example is: one solid red circle, two open blue wiggles, three speckled green rectangles. But one solid red oval, two solid red ovals and three solid blue ovals is not a set, because two cards are red and the third is blue. Figure 1 shows an example of the game, which included the example of the one, two and three red solid ovals (the reader may try to find at least four more sets to get a taste of the game).

How do people play Set?

One common experience of Set players is that there are easy and hard sets. Some sets seem to pop out, while

[1] Set is a game by Set Enterprises (www.setgame.com)

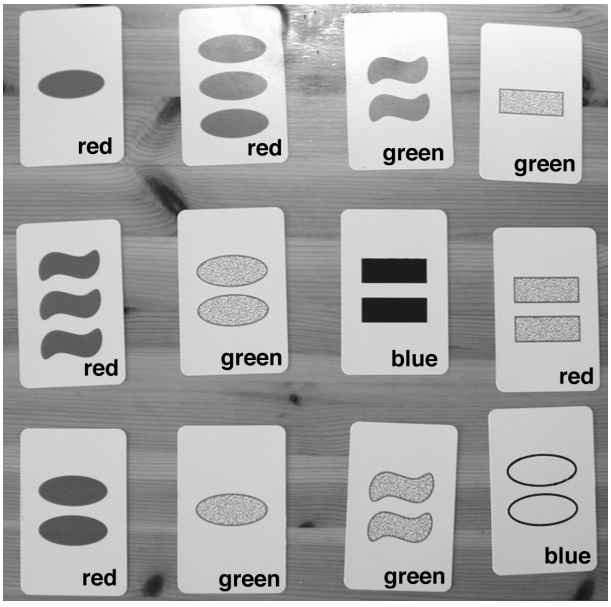

Figure 1: Example of the game Set. The color names are not part of the cards, but are added for clarity.

other require extensive search to find. Easy sets are the sets that are perceptually similar: the sets in which three out of four attributes are the same (four out of four is impossible as there are no identical cards in the game). The hard sets are the sets in which all four attributes have different values. More precisely, we have three levels of difficulty: three, two, one or zero attributes the same. Our first hypothesis about human behavior is that it will take longer to find a set as it is harder.

Our second hypothesis, based on extensive Set experience of some of the authors is that novices and experts do not differ much in performance on the easy sets. Experts mainly excel at harder sets. A preliminary explanation might be that easy sets are perceptually obvious sets, and can be identified because they "pop up". If finding easy sets is really a purely perceptual issue, one would expect little progress through learning.

Experiment

Twenty Set problems were presented to eight participants. Each problem consisted of twelve Set cards, in which a set had to be discovered. Only one set was present in each problem, and these sets varied in their level of difficulty: five of each level. The problems were presented in random order to the participants, and they had to work on a problem until they found the set or until they had searched for 300 seconds.

The eight participants were all undergraduate students of the University of Groningen. Four of them had little or no experience in playing Set, and made up the beginners group, and four were experienced players, forming the expert group.

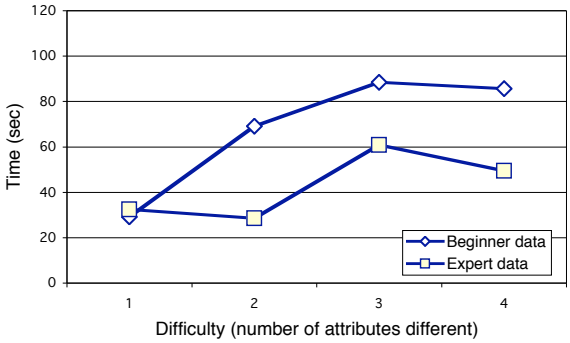

Figure 2: Results of the experiment. Time to find a solution is shown for both beginners and experts, and for the four levels of difficulty.

Figure 2 shows the results of the experiment. If a participant hadn't found a set within 300 seconds, this was counted as 300 seconds in the results. The results confirm our prior expectations that problems with more attributes that have different values are harder to find. Harder set problems take more time to solve, both for experts and beginners. Experts are faster in general, but not on the most easy problems, where experts and beginners are about equal, confirming our second hypothesis.

Strategies in playing Set

Set players report that they use the following strategies in playing the game.

The first strategy is a general method to find sets. In order to find a set, you first select two cards, and based on these two cards you determine what the third card should look like. For example, if the first card is one red solid oval, and the second card two red solid rectangles, then the third card has to be three red solid wiggles. Once the third can is determined, one has to check whether it is actually present on the table. If it is, a set can be announced, if it isn't, search is restarted.

The second strategy is only applicable in specific situations. If there are many cards with the same attribute value, for example, eight out of twelve cards are blue, it is a good strategy to search for a set that consists of blue cards. In Figure 1 six cards are solid, so it might be a good idea to look for a set in which solids.

The third strategy mirrors the second strategy: if there is only one card with a certain attribute value, for example a single red card, then search for a set with that card. In Figure 1 there is only one card with open symbols, and this card is indeed part of at least three sets.

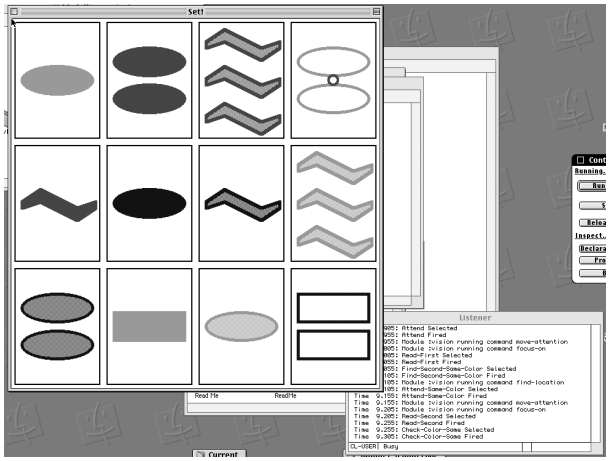

Figure 3: Impression of the ACT-R model interacting with the game

A model of playing Set

The model of Set is based on the first strategy: choose two cards and predict the third. The other two strategies will prove to be unnecessary to explain the data, although they might make a more believable player eventually. We will explore this aspect in the final application.

The model is implemented in ACT-R 5.0 (Anderson & Lebiere, 1998), using the ACT-R/PM to interface the model to the experimental task. In order to model the task we implemented a new visual object to hold the representation of a card: its color, shape, filling and number. The following aspects of ACT-R 5.0 are relevant for the present model:

- ACT-R has several perceptual-motor modules that communicate through buffers with central cognition.
- Declarative memory is also considered a module. A certain fact can be requested, and this fact will appear (when present) in the declarative memory buffer (usually called the retrieval buffer).
- These modules operate asynchronously, so their actions only have to be initiated by central cognition. This allows for limited parallelism: while an eye-movement is in progress, a request can be made to declarative memory for a certain fact, after which the results of the eye-movement can be collected.
- All operations are controlled by production rules. It is determined which production rules match the current contents of the buffers, the best one is chosen and its actions (buffer changes and new buffer requests) are executed.
- Apart from buffers connected to modules, there is a buffer that holds the current goal of the system.
- New rules are learned by production compilation (Taatgen & Anderson, 2002). Two rules that fire in sequence are combined into a single rule by substituting the intermediate retrieval into the rule itself. This creates faster but more specialized rules. As a declarative memory action is removed by this process, it may also enable parallelization of previously serial actions (Lee & Taatgen, 2002).
- ACT-R provides predictions for all the timing aspects of the model, including eye-movements, action times, time to retrieve from memory, and so on.

The model operates in the following manner. We assume the model of a beginner. Figure 3 gives an impression of the interface the model interacts with. This interface was also used in the experiment.

1. A random card is attended out of the set of twelve available cards.
2. The attributes of this card are stored in the goal.
3. A second card that hasn't previously been attended is attended. This card is preferably one that is similar to the first card. If there are no unattended cards left we return to step 1. The attended card is left in the visual buffer. So by the end of step 3, the first card is in the goal buffer and the second card is in the visual buffer.
4. The model makes a request to declarative memory whether the selected combination of two cards has been tried before. The model doesn't wait for the results, but instead starts to make a prediction for the third card.
5. The model starts making a prediction for the third card. This means that for each of the four attributes, based on the contents of the goal and the visual buffer, the predicted value is copied back into the goal buffer. For example, if the color in the goal and in the visual buffer are both red, the third card also has to be red. If the color in the goal buffer is red, and in the visual buffer is blue, the color of the third card has to be... But wait, we have to determine the third color! That means a retrieval from declarative memory is needed, but we are still waiting for declarative memory to come up with our request from step 4. So as soon as we hit upon a attribute where the goal and visual buffer are unequal (and there is at least one such attribute), we have to wait.
6. Declarative memory either has retrieved that we have tried the first two cards before, in which case we return to step 3 to select another second card, or returns a failure (we haven't tried the two cards before or we forgot that we tried them), in which we can proceed with step 7.
7. Finalize the prediction of the third card. Declarative memory is available again so we determine the third value of an attribute if needed.

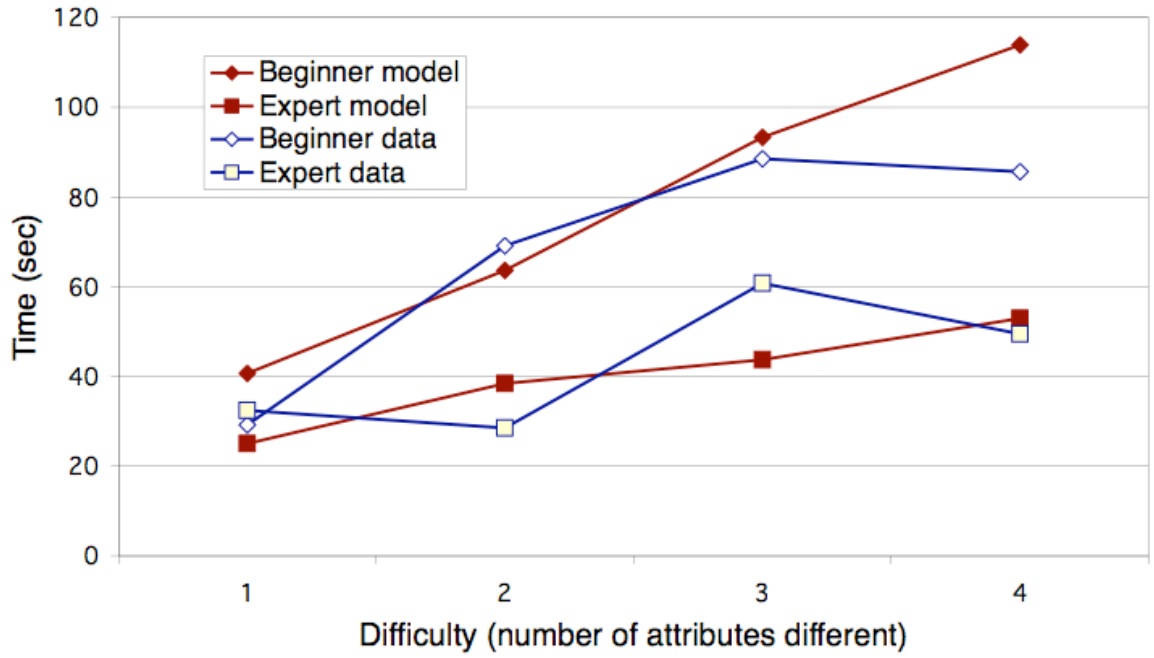

Figure 4: Results of the model

8. Scan the cards to see whether the predicted third card is actually present. If it is, announce a set has been found, otherwise return to step 1.

As this description shows, declarative memory is at some point the bottleneck in the process. This bottleneck is only present in the case of unequal attribute values. This is the explanation why equal attribute values are more attractive to scan for first, and easier to find.

The expert model

The above model is of a beginner. An expert player might have discovered more strategies in the mean time, but in our model we will only use the production compilation mechanism to speed up the processing. As mentioned in the ACT-R overview, production compilation combines two rules into one while substituting the retrieved fact into the rule. The beginner model has the following two rules:

IF the color in the goal is *val1* and the color in the visual buffer is *val2*
THEN send a retrieval request for a value that is different from *val1* and *val2*

IF the retrieval buffer contains *val3*, different from *val1* and *val2*
THEN put *val3* in the goal

Suppose these rules retrieve the fact that red is different from blue and green, then the learned rule is:

IF the color in the goal is blue and the color in the visual buffer is green
THEN put red in the goal

The attractive aspect of the rule is that it no longer requires a retrieval from declarative memory, which makes it faster than the original rules. But there is an additional bonus: the waiting in step 5 is no longer necessary, as the third card can be predicted without accessing declarative memory. So the expert can skip step 7, and never has to wait in step 5.

This also explains why experts are mainly better at the hard sets: with the easy sets most attribute values are equal, on which the expert has no advantage. But in the unequal attribute values he (or she, local lore has it that women are better at set than men) excels.

Figure 4 shows the results of the model, compared to the human data. All ACT-R's parameters were left at their default values, except the latency factor, which was set to 0.5. Although the match is not exact (the data is fairly noisy with only eight participants), the two hypotheses are captured very well.

Implementation in a Game

We used the model to implement a computer game, where the human player can play the game against ACT-R. In order to explore several possible opponents,

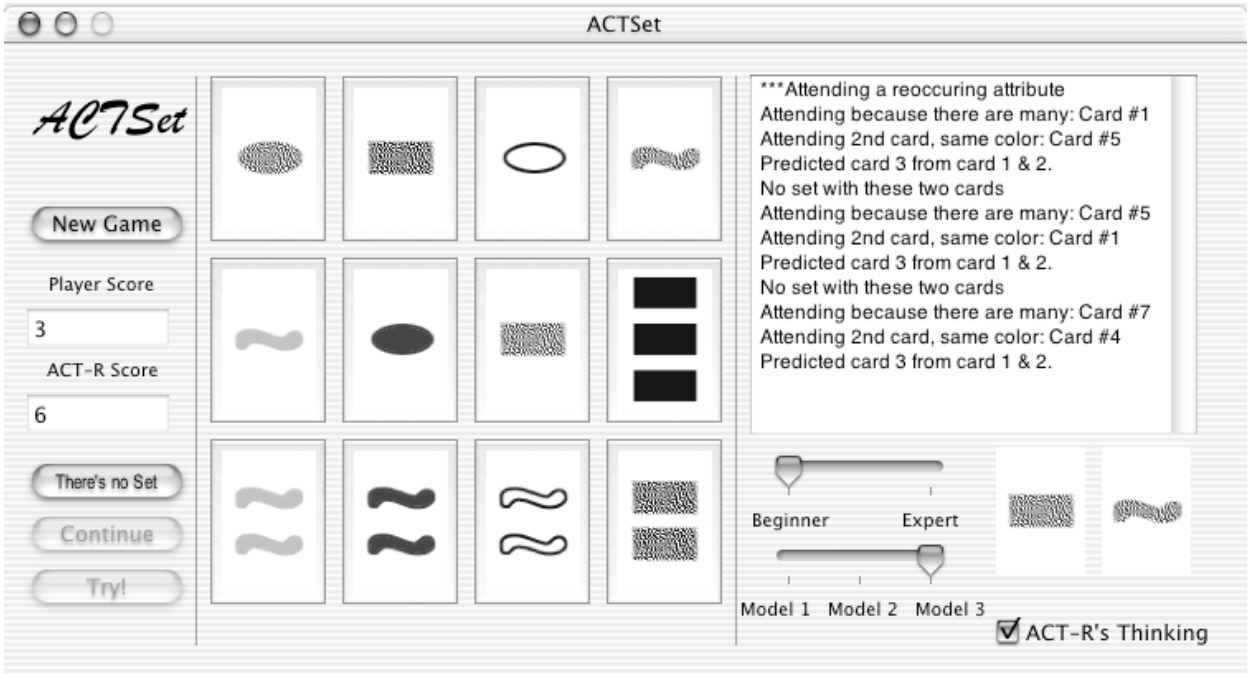

Figure 5: The ACTSet game application. Sliders control the game difficulty and the model that is used. The model's behavior can be studied by ticking the "ACT-R's Thinking" checkbox. In the figure, the model is about to announce that it has found a Set.

the application has three different opponents, each with a beginner and an expert setting. Figure 5 shows the final Application.

All three models are similar in their reaction times, about 45 seconds for the beginner setting, and 35 seconds for the expert setting.

Model 1: Fixed time

The first opponent isn't really based on a model, but instead on the average time it takes to find a solution. In the beginner setting the model takes 45 seconds to find a solution, and in the expert setting 35 seconds. These times are slightly faster then the averages in the experiment, reflecting the fact that there is usually more than one Set on the table. This is the kind of opponent that is uninteresting to play against, so we expect human players to favor the more human-like players based on the ACT-R model.

Model 2: Predicting the third card

The second opponent is based on the ACT-R model described earlier. The model selects two cards and tries to predict the third card, and also checks whether the combination has been tried before. The expert opponent can do these two things in parallel, while the beginner opponent has to serialize retrieval and prediction. The time it takes to perform all these actions are taken from the ACT-R model.

Model 3: Extension with strategies

Although the ACT-R model is sufficient to explain the data from the experiment, we felt that the additional strategies we identified should also be part of a model. The third model includes the second and third strategy. Whenever there are six or more cards that share a certain attribute value, the model tries to find a set within only these cards. Instead of selecting a random first and second card, cards with the identified attribute values are selected.

The third strategy, looking for a card with a unique attribute value, is used by the expert version only. When there is a card with a unique attribute value, this card is selected as first card, and combinations with all other cards are examined until search moves on to other combinations.

Evaluation

In order to test the application, we asked seven volunteers to evaluate the application. All volunteers had extensive experience in playing Set. In a version in which the model numbers were randomized and the ACT-R trace was disabled, they had to interact with the different models and were asked the following three questions:

- Do you think the computer behaved like a human player?

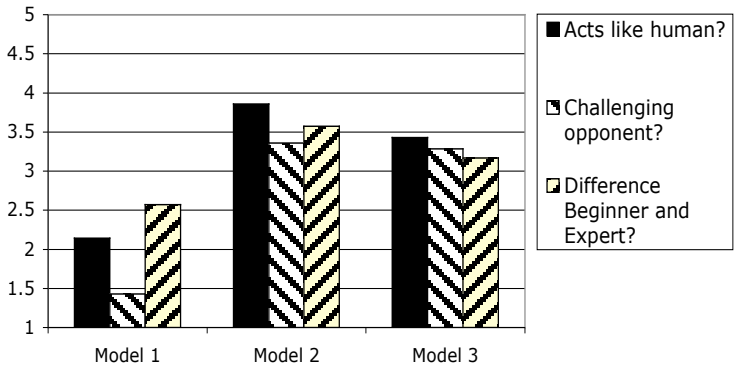

Figure 6. Results of the evaluation of the application: the three questions for each model are rated on a 5 point scale.

- Do you think the computer is a challenging opponent?
- Can you notice a difference between the Beginner and Expert settings of the model?

On each of these questions they had to answer on a five-point scale, 1 meaning "fully disagree", and 5 meaning "fully agree". They were also asked to guess what strategy the computer used. The volunteers could freely interact with the program for an hour, or until they thought they had seen enough.

Figure 6 shows the results of the evaluation. Model 1, the fixed-time opponent, was clearly rated lowest on all three questions. It was judged to be the least human, despite the fact that only one of the volunteers correctly guessed the model waits for a fixed time. It was also rated as the least challenging opponent, despite the fact that all three models are on average equally fast. Some volunteers did notice that model 1 often selects "unlikely hard sets", while not seeing easy sets right away. Judgments on model 2 and 3 are more similar, but notice that model 2 is rated as more human-like than the more advanced model 3. It is not unlikely this is due to the fact that the heuristics causes the model to more often miss obvious sets that are not noticed by the heuristics.

Discussion and Conclusion

Playing a game against an opponent who never makes a mistake is usually no fun. Computer opponents can usually only be beaten because their strategy is too rigid, and the trick is to exploit the weak spots. The ACTSet application presented here provides an attractive computer opponent precisely because it provides an opponent that is fallible. A common experience in Set is that when an opponent finds a Set, you are amazed by the fact that you haven't seen it while it was so obvious. The ACT-R opponent is the same: sometimes it is surprisingly fast at finding a Set, but sometimes it misses the obvious, and allows for the human opponent to take charge.

The application demonstrates how cognitive models can have a useful role in the design of computer games. They do not provide a way to design super-intelligent computer opponents, but they do provide a means to create human-like opponents, including reaction times, errors and maybe the occasional slip of attention.

It is not yet clear whether the model with the extra strategies will provide for a more interesting opponent. The current model 3 is even slightly worse compared to the more simple model 2. It is not unlikely only a certain level of accurateness is needed, after which additional complexity will not be noticed by the player, and might even make a less robust opponent.

The model also demonstrates a new example of an interesting modeling aspect: the fact that expert behavior is characterized by the ability to do more things in parallel. The general idea of skill acquisition discussed in Taatgen and Lee (in press) and Lee and Taatgen (2002) is that declarative memory is the bottleneck. Production compilation can eliminate the bottlenecks as long as the information that is compiled into the rules is general enough. In the Set model this is also the case: information about what the third attribute value is given two other values is stable, and can be compiled into rules.

Acknowledgements

This research was supported by NWO grant 634.000.002 (I2RP)

Downloading software

The ACTSet program requires Mac OS X to run, and can be downloaded from:
http://www.ai.rug.nl/~niels/set-app

References

Anderson, J.R. & Lebiere, C. (1998). *The atomic components of thought*. Mahwah, NJ: Erlbaum

Lee, F. & Taatgen, N.A. (2002). Multi-tasking as Skill Acquisition. Proceedings of the twenty-fourth annual conference of the cognitive science society. Mahwah, NJ: Erlbaum.

Taatgen, N.A. & Anderson, J.R. (2002). Why do children learn to say "Broke"? A model of learning the past tense without feedback. *Cognition*, *86*(2), 123-155.

Taatgen, N.A. & Lee, F. (in press). Production Compilation: A Simple Mechanism to model Complex Skill Acquisition. *Human Factors*.