Predictive ACT-R (PACT-R)

Using A Physics Engine and Simulation for Physical Prediction in a Cognitive Architecture

David Pentecost[1], Charlotte Sennersten[2], Robert Ollington[1], Craig A. Lindley[2], Byeong Kang[1]
Information and Communications Technology[1]
Robotic Systems and 3D Systems[2]
University of Tasmania[1]
CSIRO Data 61[2]
Hobart, Tasmania
Australia
email: davidp12@utas.edu.au
email: charlotte.sennersten@csiro.au
email: robert.ollington@utas.edu.au
email: craig.lindley@csiro.au
email: byeong.kang@utas.edu.au

Abstract **Advanced Cognitive Technologies can use cognitive architectures as a basis for higher level reasoning in Artificial Intelligence (AI). Adaptive Control of Thought – Rational (ACT-R) is one such cognitive architecture that attempts to replicate aspects of human thought and reasoning. The research reported in this paper has developed an enhancement to ACT-R that will allow greater understanding of the environment the AI is situated in. Former research has shown that humans perform simple mental simulations to predict the outcomes of events when faced with complex physical problems. Inspired by this, the research reported here has developed Predictive ACT-R (PACT-R), based upon integrating a three dimensional (3D) simulation of the AI's environment to allow it to predict, reason about, and then act on, what is happening, or about to happen, in its environment. Here, it is demonstrated by application in an autonomous squash player that the predictive version of ACT-R achieves significantly improved performance compared with the non-predictive version.**

Keywords- Cognitive Architectures; ACT-R; 3D Simulation.

I. INTRODUCTION

What do you do if you are asked to catch a ball that has been thrown in the air? You make a quick estimate of its trajectory, predict where you need to be to intercept it, and then move to that location. What about if it is going to bounce off a surface? Although there is now a little uncertainty, if you don't know the properties of the ball and surface, it is, nevertheless, not much more difficult to make a good enough prediction and correct for any errors after the bounce. What about if the ball has to bounce several times before you reach it? Now, you are more likely to start looking at the likely chain of events that will occur to predict the outcomes.

How could a cognitive robot – that is, a robot endowed with deliberative problem-solving – track and interact with a fast moving ball or object in a complex environment? How could a robot interact or take actions in a dynamic situation?

Artificial Intelligence (AI) in robotics commonly uses either an algorithmic approach, that is, a custom solution to a specific problem [1], or subsumption-like architectures that react to the world as it is perceived [2]. The algorithmic approach is effective for well-understood problems with little variation, but it is not so good at responding to the unexpected. Subsumption follows a 'stimulus and response' model. It is good at dealing with immediate problems, like avoiding obstacles, but can be lacking when it comes to a multi-stage mission that may require evaluation and decision-making over several alternative sequences of actions. Cognitive architectures have been proposed as an alternative that could be more suitable for accomplishing missions that require sequences of decisions, rather than more purely reactive associations between sensor inputs and motor outputs.

The American Physiological Association defines cognition as, "Processes of knowing, including attending, remembering, and reasoning; also the content of the processes, such as concepts and memories." Cognitive architectures are based on theories of how the human mind reasons to solve problems. They are used to create AIs based on, or inspired by, human cognitive processes that work through problems in a systematic way [3]. They are based on a Computational Theory of Mind, which holds that the mind works like a computer, using logic and symbolic information to work through, and solve, problems. Symbolic information is, in a programming context, a textual/verbal approach to representing knowledge in a way that is abstracted from sensory data, since the relationships between words and their referents are conventional. This abstraction supports potentially complex symbolic reasoning processes, but omits much detailed information about objects and phenomena that the symbols refer to in a given context.

Hence cognitive architectures, like other approaches to AI, have their own limitations. For example, they are similar to expert systems [4][5][6] in using facts and production rules that require a human expert to create. They are strong at

 ISBN: 978-1-61208-462-6

symbolic reasoning with logic, but the ontological status of symbols within human cognition is unclear [7], and the biological foundations of human cognition are very different from the nature of expert systems and formal logics [8]. In particular, expert systems and formal logics are technologies, i.e., inventions of human cognition, rather than its basis. They may, nevertheless, be useful and even powerful representations of some human capabilities that are based upon much lower level biological mechanisms.

An aspect of human cognition that is not captured in most cognitive architectures is *simulation*. Imagination, and the use of imagined visualisations, constitutes a conscious result of simulation within human cognition. An example of the use of simulation in an artificial cognitive system is the Intuitive Physics Engine (IPE), which uses simulation to understand scenes [7]. This method uses a fast approximate simulation to make a prediction of the outcome of a physical event or action, like the toppling of a stack of blocks.

In synthesizing a world, simulation provides a cognitive system with the richness of a sensed world, with far more detail than that which can easily be captured in higher level symbolic world descriptions alone. Simulating a 3D world and aspects of its physics involves using mathematical models of world structure, kinematics, dynamics and object interactions in which complex behaviours can be synthesized from a relatively small set of structural and physical equations. The quantisation of space and time in a simulation can be represented, e.g., to double floating point precision, resulting in an extremely large space of possible simulated world states and histories. The level of abstraction involved in declarative or symbolic representations is usually much higher than a simulated world state description, since it is expressed at a level suitable to specific decision processes, meaning that many simulation states can be compatible with a single declarative representation. That is, a declarative statement can provide a succinct and abstracted representation of a large set of world state denotations. For example the first order predicate '*is_above*(A,B)' can apply to any object in a simulation that is above another object. But to represent all of those possible individual denotations (every possible situation and variation of positions in which one object is above another) declaratively would be practically impossible. The declarative level of decision processing can be linked to the simulation state, e.g. via spatiotemporal operators linked to the simulation structure, such as testing for the relative 3D positions and sizes of objects A and B as a basis for assigning a truth value to the statement '*is_above*(A,B)'. Hence there is a useful balance between what can be represented and reasoned about most effectively using declarative representations, and the large number of potential states having small differences represented by a simulation. These are complementary modeling methods. This paper describes an experiment designed and implemented to further test the theory that simulation is a powerful component of cognition. The motivating research question asked was: "How can simulation and prediction improve decision quality in a cognitive architecture?" In the experiment designed to address this question, a predictive module was added to a cognitive architecture, and the performance of the predictive and non-predictive versions of the architecture were tested for controlling automated players of a virtual game. The predictive module used a 3D physics simulation engine to model the environment of an embodied AI, so that it could function in a dynamic situation without explicit coding of decision rules for all possible interactions in the environment. The simulation engine mathematically models interactions with the environment so that the cognitive module can handle physical events and actions with a reduced and simplified rule set.

An existing cognitive architecture, Adaptive Control of Thought – Rational (ACT-R) [10][11][12], was chosen for the research and extended with a novel predictive module. Two virtual robots were implemented to play a competitive game of squash (Figure 1). Squash is a racket and ball sport played in an enclosed room between two players. It was chosen because it provides both a physics challenge (tracking and hitting the ball), and a cognitive challenge (playing a good tactical game to out-manoeuver an opponent).

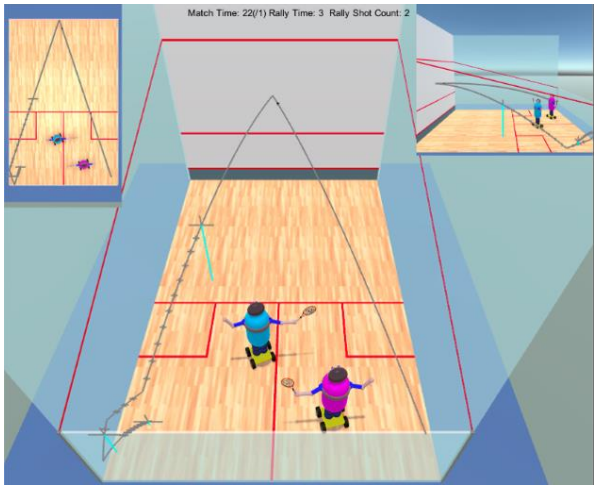


Figure 1. Squash Simulation showing AI controlled players and ball path (grey track).

Squash is a racquet sport played in a closed room between two players. The ball is free to bounce around the walls, and a player is free to hit the ball against any wall as long as it reaches the front wall before its second bounce on the floor. The opponent also has to reach the ball and play a shot before the second bounce.

The game has been described as physical chess, since it is both physically demanding and highly tactical. The physical challenge is a result of the continuous explosive acceleration needed to react to, and retrieve, an opponent's shot.

The tactical element of the game plays out in the shot selection and how this can be used to apply pressure to the opponent. When deciding when and where to hit the ball the player is faced with many choices. Do they take the ball early before it reaches a wall? Do they wait and give themselves more time to play a better shot, but also give the opponent more time to move to a stronger court position? Is a shot to the front of the court the right shot? It puts the opponent under

 ISBN: 978-1-61208-462-6

more physical pressure, but if they reach it with a bit of time to spare it opens up a lot of attacking shots.

Squash is also a game of angles, much like a real-time game of snooker. Judging and playing the angles is an important part of the game.

Using squash as the test scenario provides a known rule set for the game and existing tactical knowledge for implementing the AI models.

Two predictive elements were added to the existing ACT-R architecture. The predictive module always provided a prediction of the ball's flight path for the purpose of intercepting and hitting the ball. A further predictive element was added that allowed the AI model to evaluate its own possible actions with a simulation to determine the likely outcome of those actions. Essentially, the model was able to ask very simple "what if?" questions about how its own actions might play out in the future. Performance change due to the ability to simulate and predict actions was the metric for answering the research question.

The cognitive models implemented included three different mechanisms for choosing shots to play during a game of squash: 1) a pure random shot selection to act as a base control model; 2) a model that used rules to implement a shot selection heuristic; and 3) a model that used simulation to predict shot outcomes before selecting a shot type.

The models were evaluated by playing them against one another. Data gathered from the squash play/simulation sessions recorded detailed information about shot selection, allowing analysis of the behaviour of the models and the effectiveness of their respective shot selection methods.

Section II, of this paper, gives some background to cognitive and non-cognitive architectures. In Section III a description of the research undertaken and methodology used is given. Section IV describes the AI modelling and how prediction was incorporated. Section V discusses the results obtained.

II. COGNITIVE AND NON-COGNITIVE ARCHITECTURES

Cognitive architectures are based on theories of how the human mind reasons to solve problems. These are AI systems based on human cognitive processes that work through problems in a systematic way [3]. They are based on the Computational Theory of Mind [13], that proposes that the mind works like a computer running a program, using logic and symbolic information, to work through, and solve, problems.

The cognitivist approach follows a rule-based manipulation of symbols, and uses patterns of symbols, as designed by humans, to represent the world [14]. A key characteristic is that the mapping of perceived objects to their associated symbols is either defined by humans, or learned in a way that can be viewed and interpreted by humans. Decisions about which actions to perform are derived by processing of the internal symbolic representations of the world.

The ACT-R cognitive architecture is described in detail below. Laird et al. describe the adaptation of the SOAR cognitive architecture to robot control [15]. For the robotic control task, SOAR was extended to include mental imagery, episodic and semantic memory, reinforcement learning, and continuous model learning; it also incorporates a simultaneous localisation and mapping (SLAM) module. SOAR includes procedural memory encoded as production rules, and semantic memory implemented as declarative associations. It uses both symbolic and non-symbolic representations. A number of architectures similar to SOAR and ACT-R are reviewed in [16]. [17] take an alternative approach to cognitive architecture for robotics, proposing a content-based approach that overcomes the symbol grounding problem by matching perception and sensor data to extensive cloud-based and annotated repositories of images, video, 3D models, etc..

Most operational robots do not use cognitive architectures. Instead, traditional robotic research and control has focused on software solutions that solve problems having well formulated solutions; this can be referred to as the *algorithmic approach* [1]. These systems are particularly suited to well-defined tasks and domains, and form a foundation for robotic capabilities. However, there is a need for higher level cognitive abilities to deal with less well defined problem solving and uncertain situations where the scope for variability is not sufficiently understood or is too complex, for the development of algorithmic solutions. It is in these situations that cognitive architectures might provide an effective solution.

The subsumption architecture is another alternative to cognitive architectures for robot control. The subsumption architecture approaches intelligence from a different perspective. Rather than rules that lay out a series of steps to accomplish a task, it uses a very sparse rule set that responds to sensor values to generate control outputs [18][19][20]. Brooks describes subsumption as a layered finite state machine where low-level functions, like "avoid obstacles", are subsumed into higher-level functions, like "wander" and "explore". Each successive layer gives increasing levels of competences. Lower levels pre-empt the higher levels, such that a robot can explore, but will avoid obstacles when necessary.

Key aspects of subsumption are: that it contains no high level declarative representations of knowledge; no declarative symbolic processing; no expert systems or rule matching; and it does not contain a problem-solving or learning module [2]. It responds to the world by reacting directly to sensor inputs, in order to generate corresponding control outputs. So in a canonical subsumption architecture, there is no inherent mechanism for problem-solving in an algorithmic way.

Subsumption can be very powerful. It is based on the concept that the environment stands for itself, i.e., the architecture reacts directly to environmental features, without a mediating representation. It is a functional architecture without being, or using, a declarative model of the external world. However, without additional features, like memory and goals, it is not as straight forward to implement a mission-orientated task as it would be in a production rule based architecture. Hence these different approaches are complementary: the concepts behind subsumption – a layered set of rules implemented as a finite state machine – are not

difficult to implement, and could be easily incorporated into other cognitive architectures.

Society of Mind proposes a theory that intelligence arises from the interactions of large numbers of simple functions [21][22]. This is not an actual architecture, but rather a theory [23] that argues against the idea that a single unified architecture or solution can account for intelligent behaviour.

A robotic AI can be created completely within a single architecture, using rules that control every aspect of the decision making process, but those architectures are not always ideal for every style of decision-making. Society of Mind theory argues for a modular approach to implementing an intelligence. Implementing simulation as an extension to a cognitive architecture, but using an external 3D engine to model the environment, follows this concept. The simulation is a separate, specialised function for solving problems in dynamic physical situations.

ACT-R is a hybrid cognitive architecture consisting of both symbolic and sub-symbolic components [24][25]. It is a goal-orientated architecture. The symbolic data consists of facts and production rules. The sub-symbolic data is metadata about facts and production rules that control which facts are recalled and which production rules are chosen to fire when multiple facts and rules are available.

ACT-R consists of a number of modules that interact through a production system that selects rules to execute, (Figure 2). Each module has a buffer, which can hold a chunk of data (a key/value pair structure) representing the current state of that module.

The matching system looks for patterns in the buffers that it can use to select a production rule to potentially fire from amongst those available. Each production rule includes a pattern that gives the conditions under which it can fire. Production rules can make requests of the modules, so they can change their own internal state.

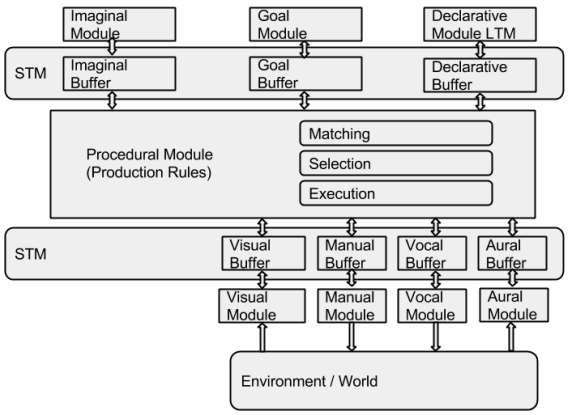


Figure 2. ACT-R structure – modules, buffers and production system.

III. METHODOLOGY

This section describes the research design, and the implementation of the prediction and simulation extensions to ACT-R to constitute the Predictive-ACT-R (PACT-R) architecture.

A. Research Design

The research consisted of developing and implementing a virtual environment for testing; developing a cognitive module that implemented the simulation-based cognition system; and developing AI models to test the system.

An ACT-R cognitive module was developed that mapped a symbolic representation of a simulated environment into the ACT-R framework. This module gave the required PACT-R functionality for interpreting and acting within the environment, as well as providing simple predictive capabilities using simulation.

The use of prediction and simulation in ACT-R was evaluated by comparing the performance of several models that each implemented different levels of prediction. The aim was to compare not only their performance, but also how easy/simple it was to model and use a predictive AI.

B. Implementation

The system implementation consisted of three components. The first was the design and implementation of a cognitive module within ACT-R. This predictive module gave models access to predictions about physical events, as well as a mechanism to take actions.

The second element was a simulation of the game of squash implemented in the Unity™ game engine. Parts of the PACT-R module were also implemented with Unity™, and communicated with the prediction module in PACT-R. The Unity™ components of ACT-R were the physics simulation and prediction engine.

The final element was modelling squash-playing AIs. Three evaluation models were developed for testing and cross-comparison.

C. Using Simulation and Prediction within a Cognitive Architecture

The research investigated the use of a physics engine to provide prediction for a cognitive architecture. The concept requires a physics engine that can model and simulate the environment of a robot controlled by a cognitive AI. The simulation provides a symbolic representation of the environment to a cognitive architecture. This gives the cognitive model (the production rules) the information it needs to understand and act within its environment.

One way of using this information is to explicitly encode rules that check for certain conditions, for example, whether an object is in a certain position, or is moving in a particular direction; or for the relationships between objects in the environment, for example, whether an object is to the left of another object [17][26]. From this, the rules can encode appropriate actions for the robot to take.

This research explored an alternative approach. Rather than using explicit rules to interpret and decide actions, a simulation of the environment was used to test actions. Figure 3 shows a high-level diagram of this approach. An environment was modelled in the physics engine that provided a squash environment and state information to a cognitive model. From the information available, the cognitive model can determine what actions might be appropriate. Rather than determining the best, with rules, it passes the choices back to

the physics engine to be simulated, which then generates a prediction of the outcome of that action. The results of each prediction are passed back to the cognitive model, which then decides which one is the most appropriate, and will therefore be used.

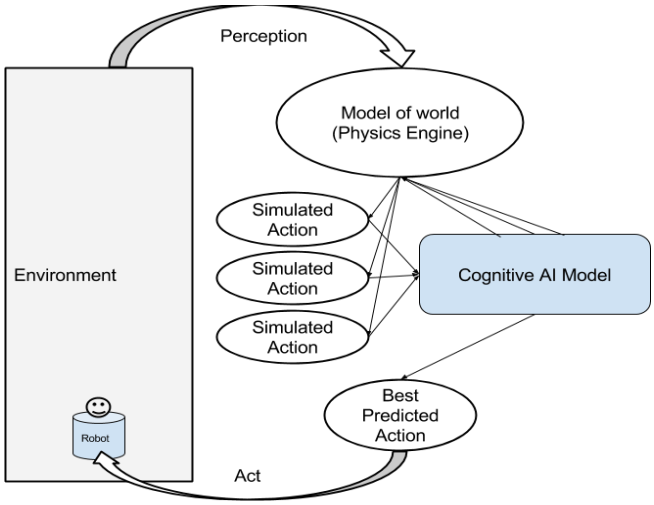


Figure 3. Overview of PACT-R concept, environment is modelled and simulated actions are tested under the control of a cognitive model.

D. PACT-R Module Implementation in ACT-R

The prediction system is implemented as an ACT-R module that both controls a robot and does a simulation of the robot's environment, for the purpose of interpreting what is happening in that environment. The module is, logically, a single system, but in the implementation it is broken into two functional parts: one residing in the ACT-R framework, and the other inside the Unity™ game engine, which includes a physics engine and also hosts the virtual world the robots exist in (Figure 4).

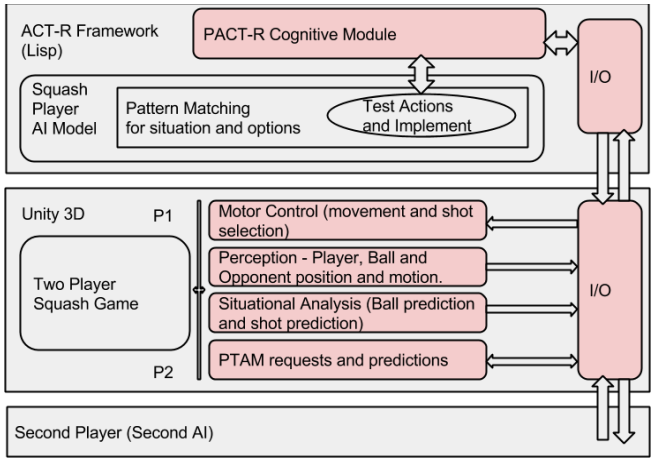


Figure 4. PACT-R (in red) within the ACT-R and Unity.

The ACT-R component of the system maintains the current simulation and prediction state for use by the AI models, while the Unity™ component of the system contains a customised physics engine that can simulate both the squash ball's path, and the outcome of shots played by the robot. The two components of the module connect via a Universal Datagram Protocol (UDP), a standard part of the Internet Protocol (IP).

For PACT-R, the cognitive module represents implicit knowledge of the sort that a squash player learns over many years. Part of this implicit knowledge is the muscle memory that allows a player to move correctly and hit a ball properly. Another part is an implicit understanding of the tactical situation. Coding this implicit knowledge into an AI model would be difficult and counterproductive. A squash player does not think about this, but rather uses it as a base to decide what they should do next. Essentially, the difference resides between 'how do you do something?' and 'what you should do?'. Implicit knowledge encodes the 'how', while the simulation provides a basis for deciding 'what'.

The PACT-R module has to work through ACT-R modules and buffers. The extended prediction module is, therefore, implemented as an additional cognitive module that provides two buffers, one that commands are sent to, and the other that gives the model access to a simplified view of the environment. The prediction module communicates with the simulation engine to both receive predictions and to request predictions based on possible actions of the AI model. Figure 4 shows the modified ACT-R framework with the additional prediction module.

IV. AI Modelling and Prediction

This section presents the outline of the AI models at a conceptual level, rather than dealing with the details of modelling them in ACT-R. Then, the implementation of the prediction module in ACT-R is presented, together with its interactions with the AI models, followed, by a description of the evaluation and analysis framework for these models.

A. Prediction Models

The simulated task, playing squash, that the AI has to perform is dynamic; the ball is in continuous motion, and can follow complex paths as it interacts with the walls and floor. Likewise, the AI's robotic avatar is moving, as is the opponent.

ACT-R is designed to look for, and respond to, patterns in information in its buffers. The buffers hold information representing both the external world, and the AI model's internal state. ACT-R can work with values and do simple comparisons, but doing complex calculations and relationships is not its forte (although it is possible to call Lisp functions if required). Ideally, the modules should do the hard work of breaking a situation into a simple symbolic representation that the AI model can reason about, by searching for patterns and relationships.

For a complex dynamic situation this may present a problem, since an AI model requires deliberation (i.e., "thinking") time. That is, it needs time to recognise a pattern and fire a production for the situation the pattern represents. For a dynamic situation, by the time a pattern has been recognised and acted upon, the situation may have already changed to something different.

 ISBN: 978-1-61208-462-6

The simulation-based module described here abstracts away the details of the environment into a simple set of relationships and events representing the elements in the scene. This abstraction is highly domain specific; in the implemented PACT-R, the abstraction focuses on the specifics of the game of squash.

For squash, PACT-R identifies three actors: *self*, *opponent* and *ball*. The module provides the AI model with information about the approximate locations of these actors within the squash court and information about what is happening, is about to happen, or what might happen. Conspicuously absent from the information is real coordinates and vectors of motion. While ACT-R can work with this sort of information, it would lead to a set of rules with a lot of spatial relationship calculations and conditions that might not be processed rapidly enough for real-time performance.

For this research, a baseline capability of the prediction module included a prediction about the immediate known ball flight path that the AI model could use to intercept the ball, at an appropriate court position, in order to play a shot. This prediction was made following the opponents shot when the ball's position and velocity could be determined. The ball's path was simulated in the physics engine, which tracked where the ball would travel as it bounced against the walls and floor. The path was calculated until it was determined that the ball would have bounced on the floor for a second time. This projected ball path was then used in the prediction module to determine locations where the player could intercept and hit the ball, based on their own movement ability.

The intercept positions were placed in the prediction module buffer used by the AI model, which allowed the models to intercept the ball without any further processing. The intercept position could have been under AI control, but this would have introduced more complexity to the modelling and introduced more independent variables to the test, making it difficult to determine cause and effect. For this reason, AI control and reasoning was limited only to the shot selection strategy.

To know where the ball and the player were within the squash court, the squash court was broken into strategic zones and all positions were given zone numbers. The squash strategy implemented in the models was also based on zones, with a limited selection of shots available for each zone. The AI models selected a shot from those available in the zone where the ball was intercepted. The zones and shots are based on squash training drills commonly used to teach players basic strategy.

B. Evaluation and Analysis

Three models were developed and evaluated. The first model was a *basic random shot selection model* that functioned as the base line to determine whether shot selection by the other models was better than random chance.

The second model was a *heuristic model* that had an explicit shot selection rule-set derived from the human developer's experience of playing squash. This model's purpose was to provide an alternative method to the prediction model.

The third model used the *predictive* features of PACT-R to test shots for their likely outcome.

In order to evaluate the performance of the three models, a large amount of automatic data gathering and logging was conducted from the virtual environment. This data gave both comparative performance of the models, and an insight into how they won or lost.

The data collected from the experiment was the result of player to player rallies between two competing AI models. The models were tested over a large number of rallies to produce data for a statistical analysis of the relative performance of the models.

For each test session the only variables were the shot selection strategies of the two competing AI models.

Test sessions consisted of two AI models (out of three) loaded into the ACT-R environment, playing against each other over a series of rallies. A rally is where the two players alternate shots until one is unable to retrieve or return the shot, and therefore loses. Data recorded included shot selection and state during the rally, and the final results of each rally. This was repeated for a fixed time (from three to eight hours) to generate a large sample set of data.

Squash starts with a serve from one player to another. For a test run, the serve was alternated so there was no bias or advantage to either model. Player 1 always started on the forehand side (right), and player 2 on the backhand. The players were ambidextrous with no advantage to either side (unlike human squash players).

V. Results and Discussion

The three models discussed here all follow the same base strategy. They have to choose from three or four shots available for the zone where the ball is to be hit. The basic model did not use any additional logic to choose a shot. The other two models tried to choose a shot that would force the opponent to have to travel the furthest to reach the ball in order to play their next shot.

A. Basic Random Shot Selection Model

The first AI model developed was a random shot selection model. This created a setup with three or four equally possible shots for each court zone for ACT-R to choose with its production rules. With no additional conditions in the rules, other than the court zone, a shot would be chosen at random from those available.

This model acted as a baseline control. It was also the only model used during development and balancing of the simulation and physics engine.

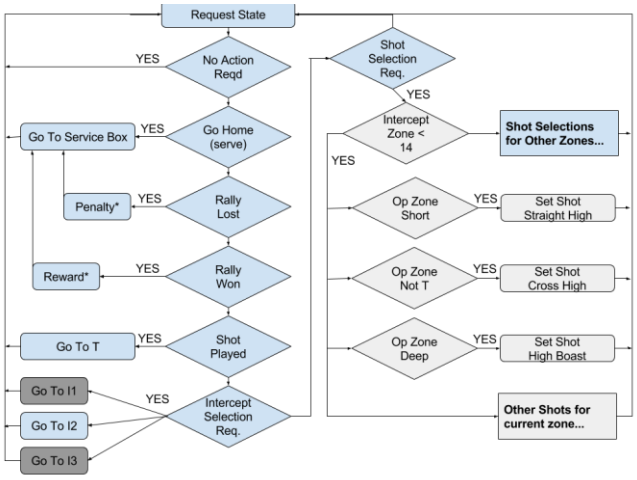
B. Heuristic Selection Model

The second model was a heuristic model that used ACT-R production rules that implemented a simple squash strategy, which tried to choose shots that would be directed to an area of the court where the opponent was not present. For example, if the opponent was deep in the court (i.e. close to the front wall of the court), it would favour a short shot; and if the opponent was on the forehand side, it would favour a backhand shot. Shot selection rules for each zone were

 ISBN: 978-1-61208-462-6

implemented using this simple strategy. In real squash, this approach is a good starting point for any human player.

Figure 5 is a flow chart representation of part of the heuristic model, although it only shows one shot selection choice, rather than the many that were required to model shots for all court zones. It should be noted that for ACT-R production rules, matching and firing does not proceed in a step-by-step fashion like a flow chart. The flow chart representation is used to show the logic, rather than the functioning of the models.




```
(p take-shot-z22-z23-StHi-OpSh
    =goal>
      ISA playing-mode
      state    2       ; play mode
    ?command>
      state free
    =predictive>    ; PACT-R module
      ISA predictive-state ; correct chunk type
      special 5     ; shot selection mode
      > intercept-zone-width 1 ; position wide
      intercept-zone-depth 2 ; position mide
      > op-zone-depth 2  ; op at front of court
    ==>
    +command>
      ISA command-packet
      req-cmd     4       ; Set Shot to play
      :req-param  51      ; Long High Straight
  )
```

Figure 5. Heuristic AI shot selection model flow chart and an example rule showing a single zone selection.

Each diamond and rectangle pair in Figure 5 corresponds to a production rule. The heuristic model consisted of 45 production rules for shot selection, plus another 5 to implement the functionality required for starting and ending a rally, and for returning to a central court position when not returning a shot.

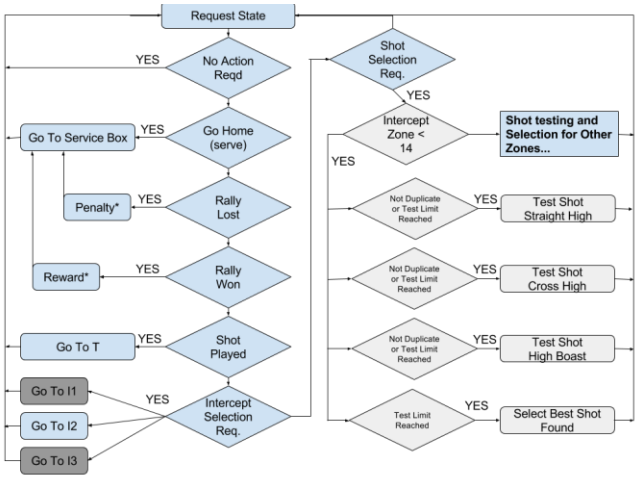
C. Predictive Selection Model

The third AI model was the predictive model. The random and heuristic models both had access to a prediction of the balls' path that they could use to determine where to go to hit the ball, and, consequently, what shots they should be playing, based on where the shot was to be taken.

The predictive model went a step further in predicting the outcome of shots the AI model might take. This was done by allowing the AI model to choose a possible shot before passing that information to the prediction module for simulating and predicting its consequences. The module would simulate how the shot would play out to predict where the opponent would be when the shot was played, and how much difficulty they would have in then retrieving it and playing a counter shot. The prediction was based on the same strategy as the heuristic model, trying to find a shot that was as far from the opponent as possible.

The prediction system has one advantage over the heuristic: as it is calculating the path of the shot under test, it sometimes found situations it could not solve for the opponent to intercept with the ball. In essence, it had found winning shots that the opponent could not return. This result was passed back to the AI, which allowed the predictive model to find, and choose, these occasional winning shots.

Figure 6 shows the prediction model as a flowchart, and a sample rule. Unlike the heuristic model's 45 rules, this model only requires 26 rules for shot selection. Each rule defines a shot to be tested for a particular zone of the court.




```
(p take-shot-z22-z23-StHi
    =goal>
      ISA playing-mode
      state    2           ; in play mode
    ?command>
      state free
    =predictive>            ; PACT-R module
      ISA predictive-state ; correct chuck type
      special 5             ; in prediction mode
      < prediction-count 4 ; more testing allowed
      - registered-shot 51 ; not already tested
      > intercept-zone-width 1 ; court pos wide
      intercept-zone-depth 2   ; and mid depth
    ==>
    +command>
      ISA command-packet
      req-cmd     5       ; Test Shot (predict)
      :req-param  51      ; Long High Straight
  )
```

Figure 6. Predictive AI shot selection flow chart and sample rule.

The predictive system works by allowing the AI model to test shots that are available to play. This allowed the prediction system to usually come up with the best shot available within the limits of the prediction resolution. Figure 7 shows the progression of the shot testing as the cyan player moves to intercept the shot. The grey track shows the ball's current path

in the top right frame. In subsequent frames blue tracks appear which represent possible shots. In the final frame the cyan player has played the best shot found which, is another straight shot down the left hand side (shown in grey again).

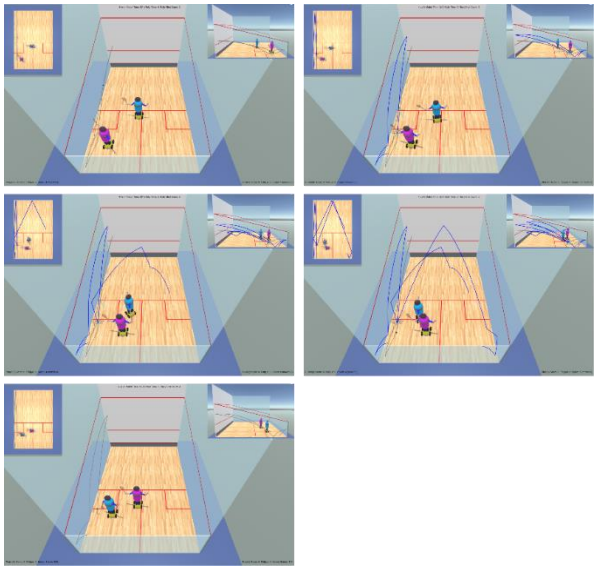


Figure 7. Time lapse of predictive shot selection showing test predictions (blue tracks) for cyan robot.

This sequence of shots takes place over a period 800ms, Figure 8 shows an abbreviated trace of the ACT-R rules firing for the sequence in Figure 7. Prediction tests are 150 ms apart, which corresponds to ACT-R's default cycle time for rule firing. The first shot tested scored the highest and is selected as the shot to play in the FINAL-SHOT-SELECTION rule fired at the end of the trace.

```
9.050 PRODUCTION-FIRED TEST-SHOT-Z22-Z23-STHI
   Testing shot 51 0
   better predicted value 2 for 51
9.200 SET-BUFFER-CHUNK SPATIAL SPATIAL-STATE45
9.200 SET-BUFFER-CHUNK SITUATIONAL-STATE45
9.250 PRODUCTION-FIRED TEST-SHOT-Z22-Z23-BODF
   Testing shot 23 1
   predicted value 1 for 23
9.400 SET-BUFFER-CHUNK SPATIAL SPATIAL-STATE46
9.400 SET-BUFFER-CHUNK SITUATIONAL-STATE46
9.450 PRODUCTION-FIRED TEST-SHOT-Z22-Z23-CRHI
   Testing shot 52 2
   predicted value 1 for 52
9.600 SET-BUFFER-CHUNK SPATIAL SPATIAL-STATE47
9.600 SET-BUFFER-CHUNK SITUATIONAL-STATE47
   …
9.850 PRODUCTION-FIRED FINAL-SHOT-SELECTION
```

Figure 8. ACT-R trace of a test and prediction sequence of rules being fired

D. Performance

Figure 9 shows the player to player performance of all three models. When playing identical models against each other the results are even, as would be expected. Both heuristic and predictive models win over the basic random selection model. The predictive model also wins over the heuristic model, with a score of 312 to 228. The binomial test p-value for this is 0.0003, showing that this is unlikely to be due to random chance.

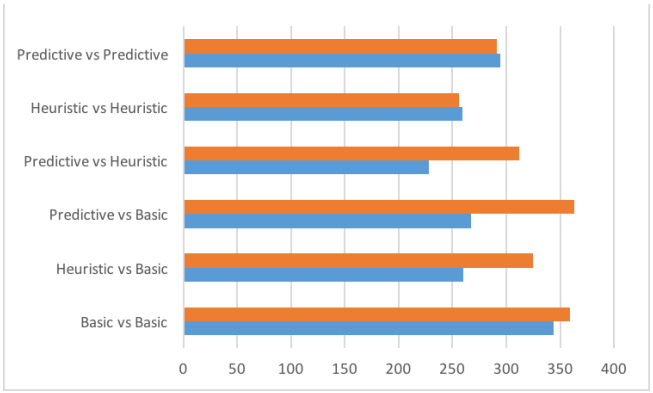


Figure 9. Head to head scores for all models over six hour duration games.

When developing the models, there was a clear advantage to the basic and predictive models over the heuristic model in the reduced number of rules required to implement the shot selection strategy. The basic and predictive models required 25 and 26 rules, respectively. The heuristic model required 45 rules to implement a simple shot selection strategy. The predictive system did have a disadvantage in the time it took to select a shot; it was not always able to complete its shot selection, and in that case it reverted to a random choice.

The three models that were developed could all play squash. The heuristic and predictive models both outperformed the basic model. The predictive system also outperformed the heuristic model, despite some limitations in its implementation.

VI. Conclusion

The research question asked "How can simulation and prediction improve decision quality in a cognitive architecture?". The results show that, within the limitations of the experiment, a predictive model – with an ability to use simulation to test its own actions to determine and evaluate their possible outcome – held a clear advantage over a model that used heuristics to test relationships between objects in a simulated scenario.

It is not, perhaps, surprising that an approach that glimpses at the future, however imperfect, would have an advantage over reasoning about a situation based only on where objects are, how they were moving, etc., in the moment. The results of the investigation indicated that prediction provided a more effective appraisal of the value of an action, without requiring detailed rules.

There is a caveat here though: the evaluation of the heuristic model was an evaluation of its specific rule set, and it could have been developed further. Its rule set was not very complicated, and it is entirely possible that with a larger rule set, and more detailed situational knowledge, it could have out-performed the predictive model. Indeed, both the heuristic and predictive models could have been developed further, to leapfrog each other in a virtual arms race.

However, there was another aspect to the modelling. The predictive model only required 26 rules versus the 45 rules of

 ISBN: 978-1-61208-462-6

the heuristic model. Not only were there less rules, they were simpler. Each rule simply stated a possible shot to test, and required no expert knowledge of how, or when, that shot might be used. In comparison, the heuristic rules required an understanding of squash strategy, and each rule had to be carefully considered as to how it would play out.

While both models could have been extended, the effort required to do so would have been considerably different. The heuristic model would require a lot of expert knowledge. The predictive model would have required only fixing some design issues and, perhaps, increasing the fidelity of the predictions. Of course, the predictive model does require a simulation engine that can predict outcomes of actions, however imperfectly. Developing the simulation does not require expert knowledge of squash either, but it does require being able to model the physics of the scenario. This is not an inconsiderable task and, even in the simple scenario used in this research, more time was spent developing the simulation than was required for the creation of the AI rule set.

VII. FUTURE WORK

The research described above only looked at a highly discrete problem, and the solution was very domain specific. The PACT-R cognitive model gave a scene description and predictions in a very squash-centric way. Continuing this methodology of creating a custom model and simulation for every scenario is time consuming, and it would be desirable to accelerate the process by finding a more generic way of describing physical relationships and actions within an environment.

It is unlikely that any solution could be truly generic. Such a solution would have to be able to model and simulate a large and arbitrary amount of the real world. Rather, a practical improved implementation of PACT-R would provide a generic framework that could be extended and adapted for specific scenarios.

Another area of ongoing research is to use PACT-R in physical robotics. PACT-R is intended for robotics and embodied AI. Taking this system into the real world presents the considerable challenge of perceiving and simulating at least a small part of the real world. For constrained situations this might not be so difficult. For example, in real-world squash, if you can detect and track the ball, it is then relatively easy to predict where it will go in the rectangular room that squash is played in. The bigger challenge would be predicting the outcome of shots, since this is not as clear-cut in the real world as it was in the simulation, since the simulated shots were simplified, and the virtual robots were able to play them more accurately than any real robot would be able to.

The research also highlighted some issues when working with ACT-R that could be an interesting topic of future work. ACT-R's reinforcement learning mechanism did not work for this task. What alternative learning mechanisms could have been used? Could some form of tagging (marking key rules in the decision process) be used so that rewards and penalties are given to the correct rules? How would the modelling need to change to make use of learning?

In modelling within ACT-R values, rules are tested with a basic set of comparative operators (>, <, =, etc.) While this is suitable for a lot of modelling, when implementing the squash strategy it would have been convenient to have been able to model in fuzzy logic, where instead of yes/no answers, cold/cool/warm/hot answers were possible. The matching would bias the rule selection, rather than simply excluding or including specific rules. Giving ACT-R a fuzzy logic matching system would allow it to work better in situations where there is not a simple black or white answer.

ACT-R also has a declarative memory system (long term memory). This was not used in this research, since it supports a different learning mechanism that did not fit with modelling squash. The mechanism is based on a principle of spreading activation, where recently used memories are more likely to be recalled, and memories that share similar content are also more likely to be recalled (this is the spreading activation). Recently recalled, or similar, memories do not apply to squash, since all shots and outcomes need to be considered equally. However, without the learning, declarative memory could have played a role in the rules in encoding combinations of zones and shots. It was not done this way, since when the decision was made to implement the models as explicit rules, reinforcement learning was still in consideration as a mechanism for improving shot selection.

If declarative memory had been used, how could it have been used, and what sort of learning mechanisms could have been applied? Could reinforcement learning be used with memories? Could there be negative and positive memories, a sort of 'positive memories' that are easily recalled, and 'negative memories' that are suppressed? These considerations may be crucial for applying simulation-based prediction in different robotic applications.

REFERENCES

[1] U. Kurup and C. Lebiere, "What can cognitive architectures do for robotics?," *Biol. Inspired Cogn. Archit.*, vol. 2, 2012, pp. 88–99.

[2] H. Q. Chong, A. H. Tan, and G. W. Ng, "Integrated cognitive architectures: A survey," *Artif. Intell. Rev.*, vol. 28, no. 2, 2007, pp. 103–130.

[3] W. Duch, R. J. Oentaryo, and M. Pasquier, "Cognitive Architectures: Where do we go from here?," *Proc. 2008 Conf. Artif. Gen. Intell. 2008 Proc. First AGI Conf.*, vol. 171, 2008, pp. 122–136.

[4] P. Jackson, *Introduction to expert systems*. Addison-Wesley Pub. Co.,Reading, MA, 1986.

[5] J. C. Giarratano and G. Riley, *Expert Systems: Principles and Programming*. PWS Publishing Co., 1998.

[6] A. Ajith, "Rule-based Expert Systems HEURISTICS," *Handb. Meas. Syst. Des.*, vol. 1, no. g, 2005, pp. 909–919.

[7] C. A. Lindley, "Synthetic Intelligence: Beyond Artificial Intelligence and Robotics," in *Integral Biomathics*, Springer, 2012, pp. 195–204.

[8] C. A. Lindley, "Neurobiological Computation and Synthetic Intelligence," in *Computing Nature*, 2013, pp. 71–85.

[9] P. W. Battaglia, J. B. Hamrick, and J. B. Tenenbaum, "Simulation as an engine of physical scene understanding.," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 110, no. 45, 2013, pp. 18327–32.

 ISBN: 978-1-61208-462-6

[10] J. R. Anderson, J. M. Fincham, Y. Qin, and A. Stocco, "A central circuit of the mind," *Trends Cogn. Sci.*, vol. 12, no. March, 2008, pp. 136–143.

[11] J. R. Anderson and C. D. Schunn, "Implications of the ACT-R learning theory: No magic bullets," *Adv. Instr. Psychol. (Vol. 5)*, vol. 5, 2000, pp. 1–34.

[12] J. R. Anderson, *How Can the Human MindOccur in the Physical Universe?* Oxford University Press, 2007.

[13] H. Putnam, "Brains and Behavior," *American Association for the Advancement of Science*, vol. Section L . 1961.

[14] S. Profanter, "Cognitive architectures," *HaupteSeminar Hum. Robot Interact.*, 2012.

[15] J. Laird, K. Kinkade, S. Mohan, and J. Xu, "Cognitive Robotics Using the Soar Cognitive Architecture," *8th Int. Work. Cogn. Robot.*, 2012, pp. 46–54.

[16] D. Vernon, G. Metta, and G. Sandini, "A survey of artificial cognitive systems: Implications for the autonomous development of mental capabilities in computational agents," *IEEE Trans. Evol. Comput.*, vol. 11, no. 2, 2007, pp. 151–180.

[17] M. Lochner, C. Sennersten, A. Morshed, and C. Lindley, "Modelling Spatial Understanding: Using Knowledge Representation to Enable Spatial Awareness in a Robotics Platform", COGNITIVE 2014, The Sixth International Conference on Advanced Cognitive technologies and Applications, Venice, Italy, 2014, pp. 26–31.

[18] R. Brooks, "A robust layered control system for a mobile robot," *IEEE J. Robot. Autom.*, vol. 2, no. 1, 1986, pp. 14–23.

[19] R. a. Brooks, "Elephants don't play chess," *Rob. Auton. Syst.*, vol. 6, no. 1–2, 1990, pp. 3–15.

[20] R. A. Brooks, C. Breazeal, M. Marjanovi, B. Scassellati, and M. M. Williamson, "The Cog Project : Building a Humanoid Robot," in *Computation for metaphors, analogy, and agents*, Springer Berlin Heidelberg, 1999, pp. 52–87.

[21] M. Minsky, R. Kurzweil, and S. Mann, "Society of mind," *Artificial Intelligence*, vol. 48, no. 3, 1991. pp. 371–396.

[22] P. Singh, "Examing the Society Of Mind," *Comput. Informatics*, vol. 22, 2003, pp. 1001–1023.

[23] B. Goertzel, R. Lian, I. Arel, H. de Garis, and S. Chen, "A world survey of artificial brain projects, Part II: Biologically inspired cognitive architectures," *Neurocomputing*, vol. 74, no. 1–3, 2010, pp. 30–49.

[24] T. Liadal, "ACT-R A cognitive architecture,", *Cognitive Science* , 2007, pp. 1–16.

[25] D. Bothell, "Extending ACT-R 6.0," 2007.

[26] C. Sennersten, A. Morshed, M. Lochner, and C. Lindley, "Towards a Cloud-Based Architecture for 3D Object Comprehension in Cognitive Robotics", COGNITIVE 2014, The Sixth International Conference on Advanced Cognitive technologies and Applications, Venice, Italy, 2014, pp. 220–225.

 ISBN: 978-1-61208-462-6