Representing Goals Modally: A Production System Model of Problem Solving in the Tower of London

Gareth E. Miles (gmiles@glam.ac.uk)
University of Glamorgan
Trefforest, CF37 1DL, UK

Abstract

GLAM-PS (Glamorgan Problem Solver) is a production system model of problem solving in the Tower of London (TOL) puzzle. It is introduced as a draft cognitive architecture that is similar to John Anderson's (1998, 2004) ACT-R, but represents system goals, long term memory and production memory modally, rather than amodally. The current paper demonstrates how GLAM-PS models problem solving on 3-disk TOL problems (a comparison with human data is also made). GLAM-PS uses representations of intended actions to control behaviour and planning utilizes the simulation of future problem states.

Keywords: Grounded Cognition; Production Systems; Problem Solving; Goal Handling

Introduction

Our ability to engage in goal-directed action is a defining characteristic of human behaviour. Different cognitive architectures have taken various approaches to modelling goal driven behaviour. Early versions of Anderson's ACT included a goal stack as part of the architecture (e.g. Anderson, 1993), more recent versions (Anderson et al., 2004) have modeled goals as alike to other memories in terms of representation and storage, but distinct in the way they are utilised (through the action of an architecturally specified goal buffer). Other prominent architectures have placed significant emphasis on the role of amodal goal representations (e.g. EPIC, Meyer & Kieras, 1997).

The current paper presents GLAM-PS a draft cognitive architecture which has been computationally implemented in the domain of knowledge-lean problem solving (specifically the Tower of London problem, a classic test of executive function commonly used in the diagnosis of Dyexecutive Syndrome). GLAM-PS is a production system architecture that represents goals modally, indeed the only cross-modal representation used is a 'system state' representation, containing details of active modal representations in each of the architectures modules.

A key feature of GLAM-PS is that it is derived heavily from current production system architectures (primarily ACT-R and EPIC). In this respect it allows for an interesting comparison between current modal-amodal architectures and a draft modal-only architecture. In the current paper these differences are examined, in particular the paper explores how executive control is handled by GLAM-PS in a classic problem solving domain. GLAM-PS can be interpreted as a simplification of existing architectures and broadly speaking is functionally similar to ACT-R / EPIC. Given this, a key question is whether GLAM-PS is a competent model of complex human behaviour and an effort is made in this paper to compare it with human data.

GLAM-PS is also an example of an implemented simulation-based grounded cognition theory of the type proposed by Barsalou (2008). A criticism of some grounded cognition theories is that they lack symbolic computational explorations that implement the somewhat abstract ideas they are based on (Dennett & Viger, 1999). Whilst GLAM-PS was *not* designed to be a computational implementation of theories of grounded cognition, it never-the-less behaves in a way that is almost entirely congruent with these grounded theories. As such, GLAM-PS provides a much-needed symbolic explication of ideas inherent in grounded cognition. The status of GLAM-PS as a simulation-based theory is most evident when it is planning ahead. GLAM-PS does this by simulating the consequences of intended actions.

The paper is structured in the following way: The first principal section outlines the theoretical assumptions that GLAM-PS is based upon, highlighting its relationship with existing cognitive architectures. The second principal section describes the computational implementation of GLAM-PS. Following this, the main section demonstrates in detail how GLAM-PS solves a 3-disk TOL problem. The paper ends with a general discussion of key features of GLAM-PS and a brief outline of how it is hoped GLAM-PS will be developed in the future.

Principal Assumptions of GLAM-PS Theory

Whilst GLAM-PS is similar to existing Cognitive Architectures, the differences of interest emerge from number of simple assumptions. These assumptions and their rationale are outlined below.

Assumption 1: Output modules (buffers in ACT-r) can be loaded with a response, without that response being necessarily executed

Assumption 2: The contents of output modules are available as an input to other cognitive processes

These two related assumptions partially specify the action of the output modules in GLAM-PS. The impact of these assumptions, and indeed GLAM-PS as a whole, is easiest to understand by relating it to ACT-r (the architecture it is primarily derived from). The key difference between the two architectures is the absence of a goal buffer/module and retrieval buffer/module in GLAM-PS. The former is a

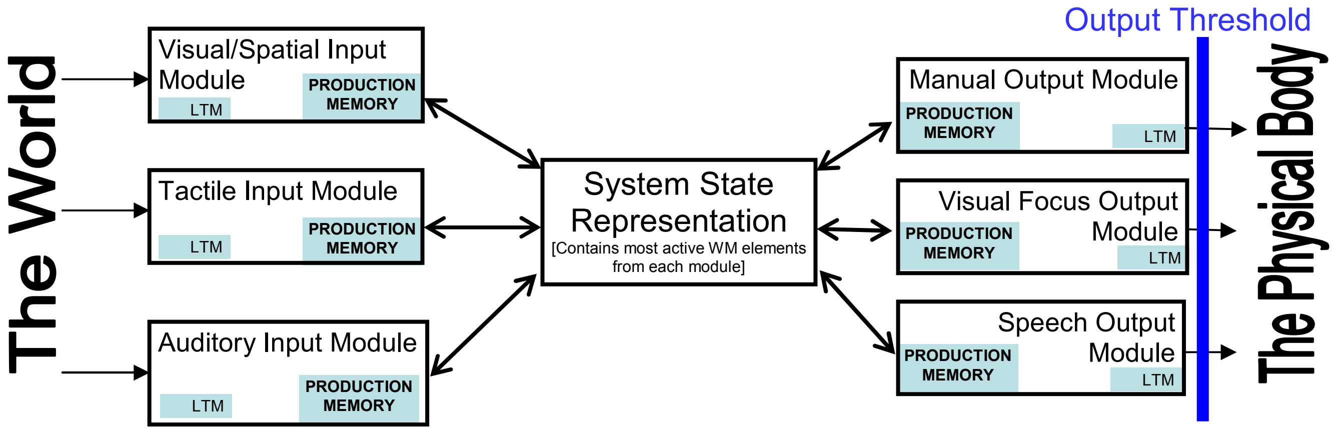

Figure 1: Diagrammatic Summary of GLAM-PS (LTM = Long Term Memory, WM = Working Memory).

direct result of Assumptions 1 and 2 (the absence of the retrieval buffer is linked to Assumption 3, see below). Crucially, Assumptions 1 and 2 allow the output modules of GLAM-PS to control action in a similar way to how ACT-r's goal buffer exerts control over action. Essentially the motor, eye movement and speech output modules are used to control and structure action in a broadly similar way to how amodal goal representations are used in ACT-r. This is best illustrated by seeing how GLAM-PS uses its output modules to control action in an example task (see the section on modelling problem solving in the TOL).

Whilst Assumptions 1 and 2 are to the author's knowledge unique to GLAM-PS (at least in the computational modelling of problem solving), a version of the third assumption is also found in many embodied theories of cognition.

Assumption 3: All Long Term Memories (incl. productions) are stored modally.

One of the primary purposes of GLAM-PS is to attempt to provide a modal-only account of complex behaviour. The combination of Assumptions 1 to 3 allows for an architecture that does not need to assume the existence of any amodal representations (cf. Harnad, 1990).

The GLAM-PS Theory

GLAM-PS (shown diagrammatically in Figure 1) is a draft cognitive architecture based on a production system formalism. As implemented (in Microsoft Visual Studio 2005) the architecture is also supported by simple modality specific semantic networks that help regulate the activation levels of elements of working memory (ACT-r adopted a similar hybrid architecture, see Anderson, 1993). Below the key features of GLAM-PS are discussed in sub-sections dedicated to Working Memory, Production Matching and Action Execution.

Working Memory

The syntax of both the working memory elements and working memory (WM) types used by GLAM-PS will be familiar to ACT-r users. However the way GLAM-PS handles WM is conceptually different from WM in ACT-r and other traditional production system architectures. In particular all WM types and elements are modality and input/output specific. Hence not only is the seen image of a square and the spoken word 'square' coded as separate WM elements, but the word 'square' heard is coded separately from the word 'square' spoken. Each WM type and element is specific to one module.

WM types are defined with a type-name, a module assignment, and then a list of slot names. For example the WM type representing seen disks in the Tower of London is defined below (left, VisIn = Visual Input):

Type: Disk, Module: VisIn	*Red_disk*
Slots: Label, Display,	*Type: Disk*
Disks_above, Disks_below,	*Label: F*
Rea, Disks, Parent	*Display: Current_state*
	Disks_above: 0
	Disks_below: 2
	Real: Yes

Individual WM elements (WME) are then specified using the format outlined in the WM type definition. The disk labelled 'F' in the TOL problems used is represented as shown above, when it is viewed.

Each WME has an activation level. This activation decreases over time, but is increased when the WME is used (e.g. in production matching) or when activation spreads from an associated WME. The level of activation can also be increased or decreased (inhibited) through the action-side of productions. The action-side of productions is also able to create new WMEs.

When a WME is referenced in the action-side of a production then other WMEs associated with this WME have their activation level increased. Associations are formed when a particular WME (e.g. A) is replaced as the most active WME within a module by a second WME (e.g. B). Everytime this occurs one unit of association is added between these two elements (e.g A and B become associated +1 unit). This, of course, dictates that inter-WME associations will be all within-module.

Activation is spread from a WME to other associated WMEs whenever the source WME's activation level is raised. A parameter is set to control how much activation is spread (e.g. currently .5 in the TOL model).

Production Matching and Conflict Resolution

The productions used by GLAM-PS are matched against the active elements in each module.[1] As an example, two simple productions, initializing TOL problem solving after reading the instructions on screen are shown below. The first searches for instructions, whilst the second represents them in the speech output buffer.

```
Prod mve_vis_focus_to_instr [P1]
 VisIn =problem
    Type: TOL_problem
>>
 +VisFoc
    Type: Search_for_target
    Target_type: written_verb_noun
    Search_Area: all_screen
    Activate: +20
```

```
Prod recode_instructions [P2]
  VisIn =instructions
    Type: written_verb_noun
    Verb: =verb1
    Noun: =noun1
  -speech =not_already_recoded
    Type: spoken_verb_noun
    Verb: =verb1
    Noun: =noun1
>>
 +speech
    Type: spoken_verb_noun
    Verb: =verb1
    Noun: =noun1
    Activate: +1
```

Buffer dependant matching of productions was explored in EPIC, and is now an established part of the ACT-R architecture (post 2000). However in GLAM-PS although production-matching is achieved in a very similar way to ACT-r and EPIC, the modules used do not act as buffers. Rather any active WME specific to a particular module can potentially be matched to a production's conditions.

Productions are matched in parallel. Conflict between productions occurs when two or more productions try to act on the same module (ACT-r handles conflicts in a similar way). This conflict is resolved by summing activation across the WMEs that were matched on the condition side of the production. The production with the greater summed activation is selected.

Productions in GLAM-PS are modal. Like WMEs, each production is tied to, and stored in, one module. Each production can only act on the module it is stored in, but is able to take input from all of the modules (including the one it is stored in).

The Action Execution Threshold (AET)

GLAM-PS as thus far described has no means of acting. The output modules hold actions ready to be executed, but this execution is not automatic or compulsory. So when will an action held in an output module be used? GLAM-PS answers this question by requiring a given action to reach a threshold level of excitation before it will be executed. Each output module's production memory not only contains productions that instantiate new actions, but also contains productions that review suggested actions and either excite or inhibit that action. In addition, productions are able to force execution (if the suggested action is judged ideal), or remove the suggested action from the module (if it is judged to be unwise). Typically though the threshold will play a role and the output module will gather evidence for and against a suggested action (in the form of exciting productions and inhibiting productions), with the action only executed when a sufficient level of excitation is reached. Executed actions are automatically inhibited to zero activation.

[1] See Figure 1 for module architecture; though note more and/or different modules are likely in future versions of GLAM-PS

Modeling Problem Solving Competence in the Tower of London using GLAM-PS

The Tower of London (TOL) is a variant of the Tower of Hanoi (TOH) that has been used in problem solving research and clinical settings for nearly 30 years. It is particularly important within Cognitive Neuropsychology, where it has been used as a test of frontal function, for instance in the diagnosis of Dysexecutive Syndrome. Example 3-disk problems are shown in Figures 2 and 3. Like the TOH, the goal is to transform the start state into the goal state, only one disk may be moved at a time, and there are three pegs. Unlike the TOH there are no size restrictions, so any disk may be stacked on top of any other disk. A key requirement of the TOL is that the shortest route (in terms of number of moves) is taken to the goal state. In essence, it is a planning task, with marked similarity to the classic blocks world task as well as to the TOH.

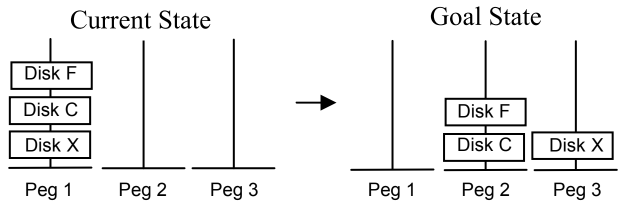

Figure 2: First three-disk Tower of London (TOL) problem (correct solution: F to 3, C to 2, F to 2, X to 3)

The best way of highlighting the impact of the unusual elements of GLAM-PS is to step through an example of GLAM-PS's behaviour on TOL problems. In the following section the key features of GLAM-PS (the action-based control mechanism and the modal long term memory) are shown in action to help facilitate understanding of how these features work in practice.

Problem Solving the TOL without Planning

In the first (and principal) example GLAM-PS is solving the problem shown in Figure 2. The first thing GLAM-PS needs to do is initalise the problem solving process. This is achieved via Production 3 (P3, see top of next page).

A verbalised instruction to solve the problem, held in the speech output module, is the catalyst of the problem solving process. The production then loads the manual output module with an underspecified disk move action (neither the disk to be moved or the location it is to be moved is specified, all GLAM-PS knows is that it needs to move a disk).

```
Prod start_to_move_disks  [P3]
  VisIn =problem
    Type: TOL_problem
  Speech =goal
    Type: spoken_verb_noun
    Verb: Complete
    Noun: Problem
  -manual =no_move_in_progress
    Type: Disk_move
>>
  +Manual
    Type: Disk_move
    Activate: +1
```

```
Prod specify_disk_for_move  [P5]
  VisIn =disk_focus
    Type: disk
    Label: =disk_id
    Display: Current_state
    Peg: =peg_id
  manual =move1
    Type: Disk_move
    Disk: [EMPTY]
>>
  manual =move1
    Disk: =disk_id
    From_peg: =peg_id
```

The next actions taken by GLAM-PS are to look for a disk. The production (P4, not shown) matches all three disks in the problem, triggering a conflict resolution process that favours the most activated disk. The problem (shown in Figure 2) is a relatively easy problem for human problem solvers, the majority of whom select the optimal move choices (61 of 87 in Miles, 1998). Though in a significant minority of cases a second pattern of moves is followed (21 of 87 in Miles, 1998; this pattern in Figure 2 would start F to 2, C to 2, X to 3). Both of these patterns stem from the same set of productions in GLAM-PS, with the different patterns a result of a different disk being the focus of visual attention initially.

The next production to execute, specifies which disk is to be moved (P5, see above). Once the disk to be moved has been specified, the next production looks at the disks location in the goal display (P6, see below)

```
Prod find_disk_in_goal_display [P6]
  VisIn =disk_focus
    Type: disk
    Label: =disk_id
    Display: Current_state
  manual =move1
    Type: disk_move
    Disk: =disk_id
>>
  +VisFoc
    Type: Search_for_match
    Target: disk
    Label: =disk_id
    Search_area: goal_state
    Activate: +20
```

```
Prod disk_requires_move [P7]
  VisIn =disk_focus
    Type: disk
    Label: =disk_id
    Display: Goal_state
    Peg: =peg_id
  manual =move1
    Type: disk_move
    Disk: =disk_id
    From_Peg: NOT[=peg_id]
>>
  manual =move1
    To_peg: =peg_id
    Activation: +5
```

If the disk is on different pegs in the current and goal displays, then the location of the intended move is specified in P7 (see above).

At this point P8 and possibly P9 will typically be instantiated (see below). P8 looks at the location of the intended move and then P9 projects the impact on the current state of the intended move (in a way very similar to simulation models).

```
Prod find_move_loc_in_current [P8]
  VisIn =disk_focus
    Type: disk
    Label: =disk_id
    Display: goal_state
  manual =move1
    Type: disk_move
    Disk: =disk_id
    To_peg: =peg_id
    Not_to: [EMPTY]
>>
  +VisFoc
    Type: Search_for_match
    Target: peg
    Label: =peg_id
    Search_area: current_state
    Activate: +20
```

```
Prod project_move [P9]
  VisIn =target_peg
    Type: Peg
    Display: Current_state
    Label: =peg_id
    Disks_on_peg: =X
  manual =move1
    Type: disk_move
    Disk: =disk_id
    To_peg: =peg_id
    Not_to: [EMPTY]
>>
  +VisIn
    Type: disk
    Label: =disk_id
    Display: Current_state
    Peg: =peg_id
    Disks_below: =X
    Disks_above: 0
    Real: No
```

P10 (see below) then compares the simulated state with the goal state. If there is a mismatch in the number of disks beneath the two disks then the intended move is inhibited. Note the production does not check the identity of the disks beneath the simulated and goal disks.[2]

```
Prod project_goal_mismatch [P10]
  VisIn =disk_goal
    Type: Disk
    Disk: =disk_id
    Display: Goal_state
    Peg: =peg1
    Disks_below: =X
  VisIn  =disk_projected
    Type: Disk
    Disk: =disk_id
    Display: Current_state
    Peg: =peg1
    Disks_below: NOT[=X]
  Manual =move1
    Type: disk_move
    To_peg: =peg1
>>
  Manual =move1
    Activation: -1
```

```
Prod execute_move  [P13]
  VisIn =disk_goal
    Type: Disk
    Disk: =disk_id
    Display: Current_state
    Peg: =peg1
    Disks_above: 0
    Parent: [EMPTY]
  Manual =move1
    Type: disk_move
    Disk: =disk_id
    From_peg: =peg1
    To_peg: NOT[EMPTY]
>>
  Manual =move1
    Activation: +1
```

A similar production (P11, not shown) checks for a match and then returns visual focus to the actual disk in the current state. There is also a production (P12 not shown) that returns visual focus to the disk in the current state without checking for a match, indeed without the need for P8 to P11 to be invoked. This helps simulate common beginner errors in the TOL where a moved disk is placed on top of a disk occupying the moved disks goal location.

Once the 'real' disk is back in visual focus with the move fully specified, the move is excited by repeated firings of P13 (see above), but only if the disk is free to move.

[2] This helps simulate an error common in human participants, that occurs when a disk is moved to a target location, but the disks below it are in the wrong order, and subsequently the participant cannot understand why the problem has not be 'solved' (as evidenced by an extreme next move latency).

If the disk is not free to move then P14 will focus visual attention on the blocking disk or disks[3] and then P15 will create a new action moving the blocking disk or disks to a location that is not the goal location of the original disk (see below). In this manner GLAM-PS is effectively creating a subgoal to remove the blocking disks in order to allow the originally intended move. Note that association will be established between the original intended move and the new move representation.

```
Prod vis_focus_to_block [P14]
  VisIn =disk_goal
    Type: Disk
    Disk: =disk_id
    Display: Current_state
    Disks_above: NOT[0]
    Peg: =peg1
  Manual =move1
    Type: disk_move
    Disk: =disk_id
    From_peg: =peg1
    To_peg: NOT[EMPTY]
>>
  +VisFoc
    Type: Objects_above
    Target: disk
    Search_area: =disk_goal
    Activate: +20
```

```
Prod subgoal_remove_block [P15]
  VisIn =block
    Type: Disk
    Disk: =disk_id
    Display: Current_state
    Disks_above: 0
    Peg: =peg1
  Manual =move1
    Type: disk_move
    Disk: NOT[=disk_id]
    From_peg: =peg1
    To_peg: =peg2
>>
  +manual
    Type: disk_move
    Disk: =disk_id
    From_peg: =peg1
    Not_to: =peg2
```

Critically the representation of the move in the manual output module is different from the representation of the originally intended move. The fact that the move is being used to remove a block allows that move to be executed without being assessed by P8 to P11 (which only assess moves designed to get disks into their goal locations). Further productions (P16 and P17, not shown) move visual focus to the peg the block is to be moved to, and then add the peg to the To_peg slot of the move. Once this is done P12 returns visual focus to the block and then P13 will execute the removal of the blocks.

When this occurs activation passes from the successfully executed representation to other associated representations in the same buffer. In this case the activation will pass to the move representation that was present when the block removal was subgoaled. In this way GLAM-PS implements an implicit form of goal stack, with executed actions triggering other actions that were active when they were first represented.

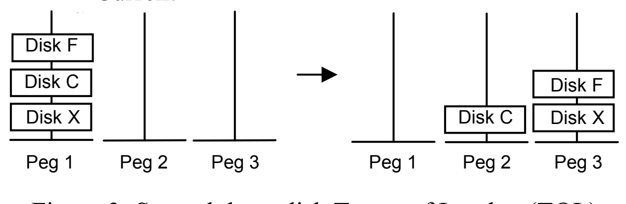

Figure 3: Second three-disk Tower of London (TOL) problem (correct solution: F to 3, C to 2, F to 2, X to 3, F to 3

Problem Solving with Planning

The 3-disk problem shown in Figure 3 cannot be successfully solved by these productions alone. P18 (not shown) can be used to generate random moves, implementing a generate-and-test style strategy. However, solving the problem in Figure 2 requires GLAM-PS to use limited planning (or be lucky). Its planning capabilities emerge from the ability to simulate a short sequence of actions and the related outcomes.[4] Previously we saw how P9 projects the impact of a move. P19 maintains the activation of a projected disk position and inhibits the disk's actual position, instead of trying to execute a possible move (it competes with P13). This allows P20 to initiate planning

```
Prod maintain_proj_disk [P19]
  VisIn =disk_goal
    Type: Disk
    Disk: =disk_id
    Display: Current_state
    Peg: =peg1
    Disks_above: 0
    Real: Yes
  VisIn =projected_disk
    Type: Disk
    Disk: =disk_id
    Display: Current_state
    Peg: =peg2
>>
  VisIn =projected_disk
    Activation: +1
  VisIn =disk_goal
    Activation: -1
```

```
Prod init_planned_move [P20]
  manual =parent_move
    Type: disk_move
    Disk: =disk_id
    From_peg: NOT[EMPTY]
    To_peg: =peg1
  VisIn =projected_disk
    Type: Disk
    Disk: =disk_id
    Display: Current_state
    Peg: =peg1
    Real: No
>>
  +manual
    Type: disk_move
    Parent: =parent_move
    Activate: +1
```

The effect of P19 is to create a simulated problem state in the visual input module. The existing productions in the model are then able to 'act' on that simulated state. The majority of the productions in the model will process simulated disk positions as well as 'real' disk positions. These productions may in turn create further simulated states. As the simulated state becomes increasingly different from the actual state the ability of P19 to maintain the simulated state will decrease. In the TOL if GLAM-PS is limited to maintaining a maximum of three simulated disk positions then a decent match to practiced human problem solving is achieved (preliminary runs of GLAM-PS on more complex TOL problems suggests that 4 or more simulated disk positions allows GLAM-PS to solve some 5-disk problems that humans often cannot solve unaided).

The handling of plan execution in GLAM-PS is currently rather inelegant. P21 (not shown) matches moves with existing parent moves (found through the parent slot) that have reached the AET (due to matching P13), it reduces the moves activation (-5), and then forces execution (+20) of any move that does not have an existing parent (this should almost always be the first move of the plan assuming it is active enough to match P21). Then, spreading activation from this initial move should be enough to trigger the next move in the plan, and so on. A production (P22, not shown)

[3] Note GLAM-PS uses the disk WM type to represent adjacent disks as perceptual groups, as well as representing individual disks. This facilitates the modelling of perceptual grouping in the TOL.

[4] GLAM-PS planning is conceptually similar to the interacting inverse and forward models used by Moller & Schenck (2008) for planning robot movement

is used to restart a plan when the activation of the next move is not greater than the AET (it looks for moves with no existing parents and increases their activation).[5]

GLAM-PS Vs Human Behaviour on the TOL

The emphasis in the current paper is on providing a detailed account of how GLAM-PS solves 3-disk TOL problems, but it is important to note that this model has been compared to human data taken from Miles (1998). The human data was taken from 231 problem solving episodes on five different 3-disk problems (involving 99 different participants). GLAM-PS was able to provide a good match for this data. Overall GLAM-PS was able to predict 209 of 231 observed move patterns (90.5%) without the need for assuming random move choices (i.e. using P18). Fourteen of the 22 unpredicted cases were on the problem shown in Figure 3 (versus 75 that were predicted on this problem). This was the hardest of the five problems used, and GLAM-PS is much more inclined to resort to random moves (P18) on this problem than on any of the other four problems, reflecting the human data.

General Disucussion

In summary, GLAM-PS provides an insightful account into problem solving in the TOL, using a draft cognitive architecture based on modal-only represenatation. Space constraints do not allow a full discussion of GLAM-PS and its relationship with existing theories of cognition, however it is important to highlight three key features within GLAM-PS that suggest interesting future directions.

The Action Execution Threshold (AET) in GLAM-PS allows actions to be represented and held, without necessarily being executed. They can then be used to control action (in a similar way to goal representations in other architectures). There are several questions about the AET that are only tentatively answered in GLAM-PS: Are there different thresholds for different modules? Are there individual differences in AET? What factors influence the AET? However, the potential usefulness of the AET construct is suggested by recent work on modelling playfulness in young children by Howard & Miles (2008). They modelled a child in a playful state as having a reduced AET (Vs the same child when not playful), thus explaining the increased behavioural fluency observed in play.

The System State Representation (SSR) used by GLAM-PS to match productions has the potential to be used to model consciousness. Intriguingly, the SSR proposal is congruent with the idea that executive function emerges from consciousness. Each module within GLAM-PS functions independently of one another, except for the influence of the SSR, which in essence binds the independent sub-systems into a cohesive whole. This idea is explored in more detail by Miles (2009).

[5] The +5 activation in the action-side of P7 ensures that plans that place disks in their goal positions will be much preferred to plans consisting of guessed moves (i.e. using P16).

Finally, it is also important to note the important role played by the speech-output buffer in GLAM-PS. There is much evidence to show that inner speech / articulation plays a role in goal-based behaviour (e.g. Saeki, 2007). In GLAM-PS the speech output module provides a necessary way of controlling complex sequences of action. In addition, it would also appear to have a pivotal role to play in any GLAM-PS based account of semantic memory.

The next stage for GLAM-PS is the modelling of human performance on 4-disk and 5-disk TOL problems. Although detailed matches to human behaviour haven't been established yet on these more complex problems, early model runs indicate that the 3-disk model generalizes to these more difficult problems. It is anticipated that GLAM-PS will eventually be used to simulate performance as well as competence in the TOL. There are also plans to extend GLAM-PS to Tower of Hanoi problem solving, Episodic Memory Recall, and Serial Recall.

Presently GLAM-PS is a work in progress. It has only been used as a computational model in one domain, the conflict resolution mechanism is underspecified, it does not currently have a learning mechanism (though one has been outlined on paper), and it is not yet suitable for comparison to detailed performance data (e.g. RTs). It is hoped these, and other, shortcomings will be addressed in future.

References

Anderson, J. R. (1993). *Rules of the mind.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of mind. *Psychological Review, 111*, 1036-1060.

Barsalou, L. W. (2008). Grounded cognition. *Annual Review of Psychology, 59*, 617-45.

Dennett, D. C., & Viger, C. D. (1999). "Sort-of symbols?". *Behavioural and Brain Sciences, 22*, 613.

Harnad, S. (1990). The symbol grounding problem. *Physica, 42*, 335-346.

Howard, J. L., & Miles, G. E. (2008). A behavioural threshold and fluency theory of play. *BPS Education Section Conference*, Milton Keynes, UK.

Meyer, D. E., & Kieras, D. E. (1997). A computational theory of executive cognitive processes and multiple-task performance: Part 1. Basic mechanisms. *Psychological Review, 104,* 3-65.

Moller, R., & Schenck, W. (2008). Bootstrapping cognition from behaviour – A computational thought experiment. *Cognitive Science, 32*, 504-542.

Miles, G. E. (1998). Reminding in a knowledge lean domain. *Unpublished PhD Thesis*, Cardiff University.

Miles, G. E. (2009). How can executive function emerge from consciousness? Evidence from a production system model. Paper to be submitted to *International Conference on Cognitive Modeling 2009*.

Saeki, E. (2007). Phonological loop and goal maintenance: Effect of articulatory suppression in number size consistency task. Psychologia, 50, 122-132.