Spatial Plans, Communication, and Teamwork in Synthetic MOUT Agents

Bradley J. Best
Human-Computer Interaction Institute
Carnegie Mellon University
Christian Lebiere
Micro Analysis and Design, Inc.
bjbest@cmu.edu, cl@cmu.edu

Keywords:
ACT-R, MOUT, Cognitive Architecture, Spatial Representation, Virtual Reality, Unreal

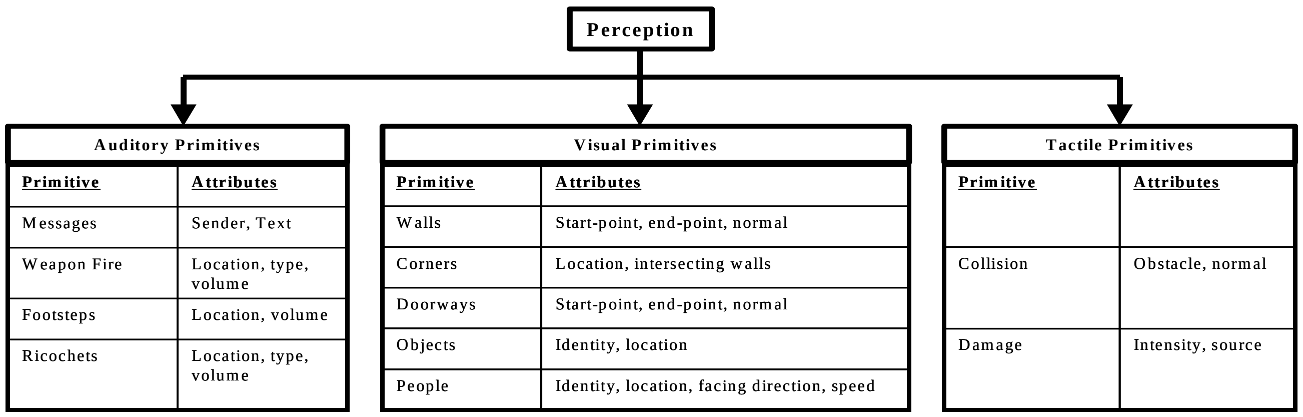
Abstract: Previously we reported on the use of Cognitive Architectures in providing realistic training opponents for virtual training in military operations in urban terrain (MOUT). That report detailed the use of ACT-R, the Unreal Tournament platform, a mapping agent for extracting cognitive primitives, and a brief description of a basic scenario involving agents within the MOUT context. This report extends the former report by describing in detail the spatial representation used by agents as well as the agents' reliance on planning, teamwork, and communication and an authoring tool to support these facilities within the ACT-R cognitive architecture development environment.

1. Using a Cognitive Architecture to Simulate Opponents in Virtual Environments

The Virtual Technologies and Environments (VIRTE) program aims to develop and demonstrate leap-ahead human-immersive technology for naval training. The goal is to train warriors for increasing complexity and chaos by supplementing and complementing live simulations using virtual and wargaming simulations and other emerging technologies. VIRTE aims to incorporate current understanding of human behavior and learning theories into systems, leverage commercially available advanced technology, evolve this current understanding into products, and then transition these products to the naval forces. In a previous paper, we introduced our effort to improve the cognitive validity of synthetic soldier entities in simulations. This effort involved the ACT-R cognitive architecture for modeling and the Unreal Tournament virtual reality game platform for a simulation environment. The current report details further progress on the project including the refinement and development of robust spatial reasoning, communication, and teamwork between cognitive agents. [4]

2. Perception in a Virtual Environment

The context an agent experiences depends completely on the senses it is provided with to perceive that context: the agent's reality is the internal representation of those senses. The ACT-R/MOUT framework currently provides for visual, auditory, and tactile senses. Some of the primitives obtained through these senses and their attributes are listed in Figure 1 below.



Auditory Primitives

Primitive	Attributes
Messages	Sender, Text
Weapon Fire	Location, type, volume
Footsteps	Location, volume
Ricochets	Location, type, volume

Visual Primitives

Primitive	Attributes
Walls	Start-point, end-point, normal
Corners	Location, intersecting walls
Doorways	Start-point, end-point, normal
Objects	Identity, location
People	Identity, location, facing direction, speed

Tactile Primitives

Primitive	Attributes
Collision	Obstacle, normal
Damage	Intensity, source

Figure 1: Perceptual Primitives

The ACT-R/MOUT framework uses architectural features to describe the visual surroundings of an agent. These visual primitives include walls, corners (intersections of walls), doorways, stairwells, and entrances. In addition there is an extensible set of object primitives including other agents and items such as weapons, ammunition, and furniture. Auditory primitives include verbal messages (encoded as text), and sounds such as weapon fire, footsteps, etc., which are localized in space. Tactile inputs are limited to sensing contact with architecture or objects in the environment. This allows the agent to, for example, feel its way along a wall, or to stop backing up when it makes contact with another player.

2.1 Spatial Representation

Many of the senses provide information about localization in space. In order to perceive, react, navigate, and plan, it is necessary for the agents to have a robust spatial representation. The agents use a representation based on human spatial cognition. Agents can represent things in two fundamental ways: where something is relative to the agent's location, or egocentrically (e.g., something is to my left); or where something is in absolute terms relative to a world coordinate system, or allocentrically (e.g., something is at a particular latitude/longitude). This combination of egocentric and allocentric representations has previously been used successfully in systems such as TAC-AIR Soar [16]. In addition, the agents' design is influenced by Frank's work on navigation and spatial reasoning in autonomous robots [7], which also borrows heavily from studies of human spatial cognition and incorporates many of the recent advances in specifying a qualitative spatial representation calculus. [5]

The egocentric representation of an item includes both the distance to the item and its relative bearing. Distance and bearing are both represented quantitatively and qualitatively; these alternative representations are complementary and provide unique advantages for processing in certain situations. Quantitative representations allow for precise reasoning about geometry, while qualitative representations provide an advantage for applying reasoning and logic to planning and decision-making.

In detail, distance is represented both as: 1) absolute distance to the target, and 2) descriptive distance to the target. Absolute distance is simply a numerical estimation of the distance. In psychophysical studies, human subjects have consistently demonstrated the ability to generate numerical distance estimates that scale linearly with increasing distance (i.e., best described by a

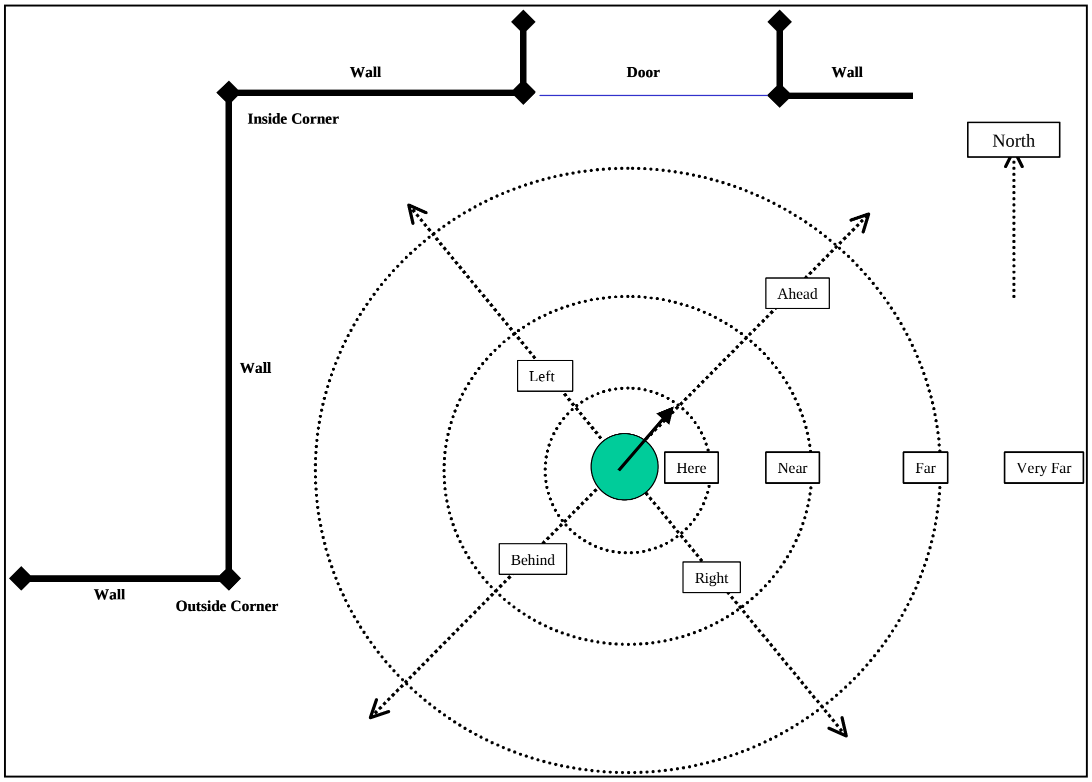


Figure 2: Egocentric and Allocentric Representations

power function with an exponent of 1) within the ranges typically encountered in settings such as MOUT [10]. A descriptive distance, in this implementation, is how distant something is relative to the current visual horizon, and ranges across “here” (within 1/8 the distance to the horizon), “near” (between 1/8 and 1/4 the distance to the horizon), “far” (between 1/4 and 1/2 the distance to the horizon), and “very far” (beyond 1/2 the distance to the horizon). Following work on qualitative spatial calculus [5], the number of levels of distance is pragmatically selected to provide an appropriate level of granularity to afford the desired behavior of the agent.

In the context of this modeling effort, the closest distance, ‘here’, is used to indicate that a particular location is within the immediate proximity of the agent, corresponding to the region described by psychological theory as ‘personal space’. The second two distances, ‘near’ and ‘far’, divide the space used for local navigation such as walking through doorways and corresponds to action space. The final distinction, ‘very far’, allows for organization of behavior relative to very distant objects and architecture, which fall within ‘vista space’ [6]. The organization of absolute distance numerically and qualitative distances logarithmically is consistent with studies indicating that children commonly represent numerical quantities using both representations [14], as well as with recent studies indicating that adults compress space at larger distances [6]. This logarithmic compression of numerical intervals provides a simple way to extend this representation to different domains by either 1) varying the relative scale (in this case, the distance to the horizon) to create larger regions corresponding to the distance labels, or 2) by extending the scale to include more levels (with each additional level doubling the distance represented).

Bearing is represented as both: 1) absolute compass bearing to target relative to current orientation (e.g., 30 degrees to the left, 5 degrees up), and 2) descriptive bearing to the target. A descriptive bearing may be “right”, “left”, “ahead”, “behind”, or any of the four intermediate bearings “ahead right”, “ahead left”, “behind right”, or “behind left”. See Figure 2 for a diagram of this. In keeping with studies of human spatial cognition indicating that humans maintain egocentric representations with much more accuracy than allocentric representations, the agents rely extensively on egocentric representations. [9]

The allocentric representation of an item includes the location of an item in the world coordinate system (in this case, x, y, and z) and its orientation relative to that coordinate system (pitch, yaw, and roll – the angles relative to the axes). An allocentric representation is particularly important in reference to maps (which are typically defined relative to some world coordinate system), and correspondingly to navigation tasks.

Though this is not immediately obvious, allocentric and egocentric representations are complementary. The egocentric representation of an object always has a corresponding allocentric representation. Many studies of human spatial cognition have been directed at how people translate from one representation to the other (e.g., map following, [9]). In general, the finding in studies of human spatial cognition is that the egocentric representation of the world is privileged, and that the allocentric representation of the world is calculated at a cost. This is also supported by computational work such as Agre and Chapman’s seminal work on AI spatial reasoning systems such as Pengi [1], which showed that situated, or indexical, representations are simpler for some calculations.

An example that demonstrates the utility of egocentric representations is aiming at a target. If player A is attempting to target a moving player, player B, the egocentric representation is extremely useful in specifying the necessary actions of player A. In this case, the absolute bearing and speed of player B is not as immediately useful – player A needs to know whether they are on target, and what rate of rotation (if any) is necessary to remain on target. This allows writing a rule for aiming such as the following pseudo-rule:

> If there is a selected target whose bearing to me is greater than 1 degree then correct my aim by turning towards the target.

The use of an egocentric representation allows authoring rules that are semantically transparent. Although it would be possible to author the same rule using an allocentric representation, the rule would lose much of its transparency. Similarly, many of the doctrinal rules for MOUT can be spelled out clearly using an egocentric representation. For example, the following describes how to employ the cross method of clearing a room:

> “When employing the cross method, two Marines position themselves on either side of the entryway. Each Marine faces into the room covering the corner of the room opposite his position. On a prearranged signal, each Marine alternately enters the room. Each Marine crosses quickly to the opposite corner while covering the half of the room toward which he is moving. Once in the near corner, he assumes an outboard kneeling position to reduce his silhouette and continues to maintain coverage of his half of the room.” [11, p. A-34]

3. Navigation

Navigation is one of the most essential tasks an agent must undertake in a MOUT environment. Navigation is accomplished through combining basic locomotive behavior with the immediate results of perception – that which is perceived at that moment – and interaction with a memory-based cognitive map of the environment.

Locomotion is simply moving from one location to another. This is a fundamental behavior that need not be attended to once it is initiated, and thus may occur in parallel with other activities. The basic behavior of locomotion involves commencing movement to a location, the continuation of that movement while not at the destination, the abandonment of that movement if an obstacle is encountered, and the cessation of movement upon arrival at the destination. In addition, locomotion can be performed in several modes: walking, running, stalking, and sidestepping while facing another direction. Receiving an order or reacting to a new threat can also interrupt locomotion.

Higher order navigational behavior involves an interaction of the cognitive map of the environment (the allocentric reference frame) with the current visual scene (egocentric cues) and memory for goals and past events (paths followed and destinations). As such, it represents a significant theoretical challenge in both cognitive psychology [9] and robotics [7].

Agents in this simulation use a simple node-link representation for rooms and pathways between them. Attacking agents build up a representation of rooms visited, as well as the episodic trace of items and other agents seen there. When moving from the current room through a doorway to a new room, the agent creates a chunk in declarative memory corresponding to that path. Defending agents, who are assumed to have intimate knowledge of the area to be defended, are given a complete representation of the rooms and pathways connecting them within a building. This allows them to fluidly and quickly choose paths for attack and escape which real defenders would have knowledge of.

Although memory for paths exists in a complete form in the defenders' declarative memories, the attackers may be forced to rely on other methods. In addition to remembering the path followed, attackers may also encode individual moves at particular situations. This is similar to the heuristic applied by some people who "retrace their footsteps" when trying to find their way. These previous moves can be actions relative to landmarks (e.g., turn left at the L-shaped hall), actions relative to an allocentric frame (e.g., proceed at a compass bearing of 90 degrees), or actions relative to an egocentric frame (e.g., turn left 45 degrees). These representations are complementary, and are typically used by people as the context allows. Landmarks are often preferred, but in a situation where landmarks are impoverished, people quickly adopt the other strategies. If going to a house in a subdivision where all of the houses look alike, people commonly depend on memory for the moves such as "turn left at the second street in the subdivision and go to the fourth house on the right". In a combat context, an allocentric frame such as that encoded in a map is often used. This is particularly useful in military situations for exchanging information about threats, destinations, and movements, since allocentric coordinates such as GPS coordinates are unambiguous, while egocentric coordinates depend on knowing the egocentric orientation of the perceiver and are therefore often less useful.

A concrete example of the interplay of egocentric and allocentric representations is the interpretation of a spot report, such as the notification of a hostile encounter. In this case, the notifying agent translates the location of the hostile force to allocentric, or world-centered coordinates (e.g., GPS location) and communicates this location to friendly forces. An agent receiving the communication determines if the location of the hostile contact is near enough, using qualitative distance to reason, to make it necessary to take action. In this case, if the hostile contact is proximate, the agent must determine the egocentric bearing to the contact using the allocentric coordinates. If the agent receiving word of a proximate threat is not facing the direction indicated by the spot report, the agent must turn to face the contact.

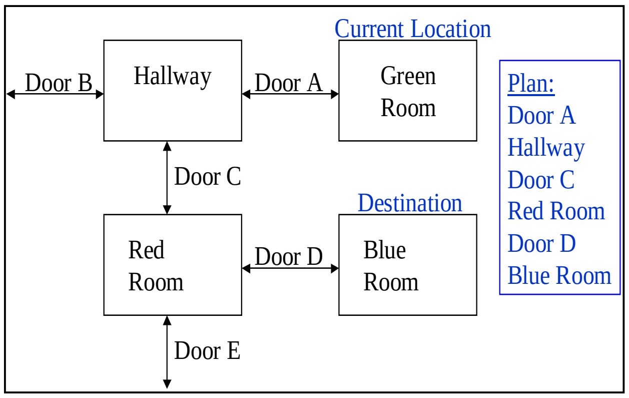


Figure 3: Node-link Representation of Rooms and Pathways and a Plan to Reach a Destination

4. ACT-R

ACT-R is a unified architecture of cognition developed over the last 30 years at Carnegie Mellon University. At a fine-grained scale it has accounted for hundreds of phenomena from the cognitive psychology and human factors literature. The most recent version, ACT-R 5.0, is a modular architecture composed of interacting modules for declarative memory, perceptual systems such as vision and audition modules, motor systems such as a manual module, all synchronized through a central production system (see Figure 4). This modular view of cognition is a reflection both of functional constraints and of recent advances in neuroscience concerning the localization of brain functions. ACT-R is also a hybrid system that combines a tractable symbolic level that enables the easy specification of complex cognitive functions, with a subsymbolic level that tunes itself to the statistical structure of the environment to provide the graded characteristics of cognition such as adaptivity, robustness and stochasticity. [2]

The central part of the architecture is the production module. A production can match the contents of any combination of buffers, including the goal, which holds the current context and intentions, the retrieval buffer which holds the most recent chunk retrieved from declarative memory, visual and auditory buffers that hold the current sensory information, the manual buffer which holds the current state of the motor module (e.g. walking, firing, etc), as well as buffers defined specially for the task. The highest-rated matching production is selected to effect a change in one or more buffers, which in turn trigger an action in the corresponding module(s). This can be an external action (movement, firing, etc) or an internal action such as requesting information from memory. Retrieval from memory functions on a similar manner. A pattern specified in a production is sent for matching in declarative memory. Each chunk competes for retrieval, with the most active chunk selected and returned in the retrieval buffer. The activation of a chunk is a function of its past frequency and recency of use, the degree to which it matches the requested pattern, plus stochastic noise. Those factors confer memory retrievals, and behavior in general, desirable “soft” properties such as adaptivity to changing circumstances, generalization to similar situations, and variability.

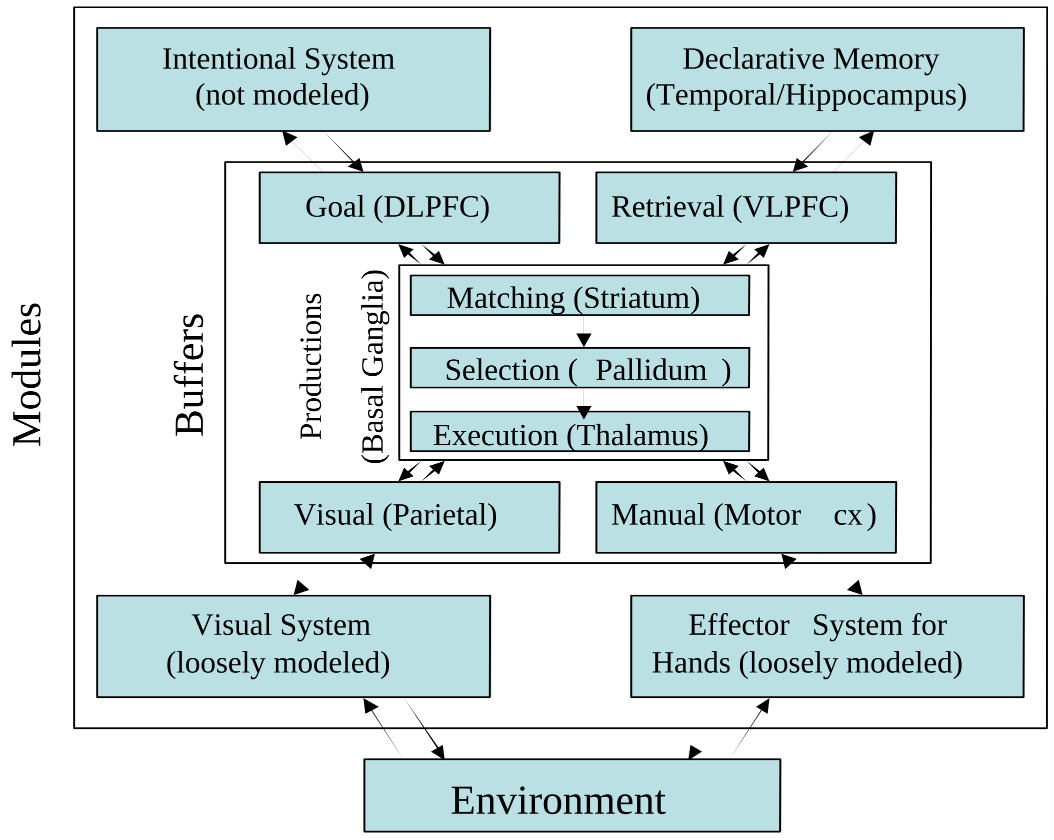


Figure 4: ACT-R Architecture

4.1 Implementing Planning and Teamwork in ACT-R

Within the framework developed for this project, a set of productions interprets the schema within the current context, leading to a literal interpretation of the schema for that context. In this way, an abstract plan plus a context results in a set of concrete actions. This allows the abstract plan to be fairly brief and vague until the agent actually selects it to be put into action. [13]

The current modeling effort includes plans for a team of two for clearing: rooms with and without doors, halls, L-corners, T-intersections, and stairs. In addition, plans are included for advancing and retreating in a leapfrog style, and for firing a defensive shot in an attempt to cause casualties immediately prior to escaping (cut and run). A sample chart of the interactions of two agents clearing an L-shaped hallway is presented below in Figure 5.

At each step of the plan, agents perform an action, communicate, or wait for a predetermined signal or length of time before moving on to their next action. In this way, the agents synchronize their actions taking turns appropriately. The plans the agents adhere to are not ad-hoc but instead come directly from doctrinal MOUT documents. Doctrine typically not only specifies how one agent should be positioned relative to another for activities such as clearing L-shaped halls but even specifies the exact language to be used in this situation. This knowledge is typically a set of steps spelled out in declarative form with the particular actions, triggers, and synchronization of action all clearly defined. Given the cookbook nature of some of these doctrinal maneuvers, we noticed an opportunity to create an authoring tool to aid in the conversion of doctrine to cognitive models. [11]

4.2 Authoring Tools

The model for the agents described here was authored like a typical ACT-R model, that is the knowledge structures, especially declarative chunks of information and production rules, were written using an abstract notation rather typical of production systems. Table 1 presents a typical example of a production rule and related chunks:

(p get-next-action	(action11
=goal>	isa action
isa action	plan 1
plan =plan	index 1
index =index	type move
type nil	argument l1)
argument nil	(action12
=action>	isa action
isa action	plan 1
plan =plan	index 2
index =index	type wait
type =type	argument go)
argument =argument	(action13
==>	isa action
=goal>	plan 1
index (1+ =index)	index 3
type =type	type end
argument =argument)	argument none)

Table 1: ACT-R Syntax of Productions (left) and Chunks (right)

The production implements the sequential retrieval of a piece of an action plan, and the declarative chunks represent some of those action pieces. While the details of the syntax do not matter, what is important is that authoring, debugging and maintaining the model is a highly specialized skill left to knowledge engineers familiar with the particular formalism in which the agent is implemented. A very desirable improvement would be to remove direct involvement with the syntax to allow subject matter experts to directly author behavioral models rather than have their knowledge extracted and codified by knowledge engineers. This would result in a faster, cheaper and easier development of behavioral models than the current, specialized, way. We are currently working toward developing an interface that would provide a first step toward that end. A simple mock up to illustrate the issues is provided in Figure 5.

The situation involves two agents, L for Leader and F for Follower, moving in coordinated fashion through a sequence of positions and actions. Their various positions are indicated by an index. Solid arrows between successive positions indicate movement. Dotted arrows indicate when an agent waits for the other to have performed an action (such as reached a position) to proceed with the next step of its plan. Dashed arrows indicate synchronized actions between the two agents. Other codes specific to the domain can be added to the graphical interface in a modular fashion.

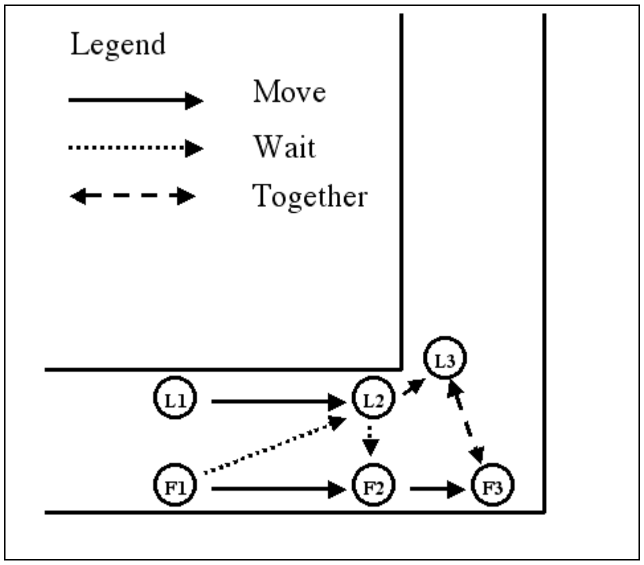


Figure 5: Authoring Interface

All those codes transform readily into a piece of the plan for each agent as encoded in declarative chunks in Table 1. Each chunk contains a number of slot. The index of the plan, **plan**, and the index of each action, **index**, can easily be supplied automatically by the interface. The nature of the action, **type**, depends on the code used in the graphical interface, e.g. a solid line would translate into a move action, etc. A list of interfaces codes and associated actions can simply be encoded into the interface for each domain. The last slot, **argument**, is an action qualifier, such as where to move, e.g. to position L2. This argument represents the most difficult part of the mapping, because obviously one does not want to encode a specific location but instead one that will generalize to similar situations (in this case, the position nearest the corner of the L-shaped hallway). Humans, even non-experts, usually understand readily using a broad common sense knowledge base the set of spatial relationships between the various positions and landmarks to generalize them across situations, e.g. to symmetrical situations. The challenge before us is to either provide in the model a sufficient knowledge base of the domain to supply those spatial relationships automatically, or to provide in the authoring interface a set of primitives to allow the subject matter experts to specify the relationships themselves. More likely, a combination of the two will result, with an iterative dialog between authoring tool and SME taking place to disambiguate the generalization of the specific plan.

4.3 Proceduralization

Plans of action are represented in the form of a list of declarative chunks (see Table 1 for an instance) each representing a particular step of action such as moving to a location, waiting for a partner, firing at a target, etc. The execution of those plans takes the form illustrated in Figure 6. Each cycle consists of a production firing requesting the retrieval of the next step (P_R), the retrieval itself (Ret), then one or more production firings implementing that course of action (P_A).

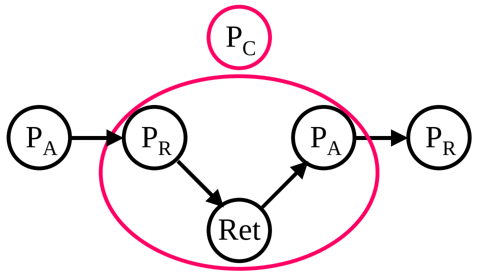


Figure 6: Knowledge Compilation

While production firings are quite rapid (usually taking about 50 milliseconds), retrieval of a chunk of information from declarative memory typically take several hundreds of milliseconds. This corresponds to a poorly trained opponent who consistently thinks about his actions rather than simply executing them. To represent a better trained opponent able to execute plans of action much more quickly and efficiently, one can take advantage of a feature of the ACT-R architecture which compiles consecutive productions, together with an intervening information request such as retrieval from memory, into a single production (P_C), specialized to the task at hand, which can then fire much more quickly than the series of interpretive steps that it replaced.

One feature of declarative retrievals is the ability to generalize to related situations based on similarities between components such as distances, angles, appearances, etc. This is quite useful in applying plans of action flexibly to situations that do not quite match the original design. We are currently working on endowing the production matching process with the same capacities for generalization to similar situations. In the current version of ACT-R, the range of application of a production has to be specified by the modeler, a practice that is neither robust or efficient, nor compatible with the use of the production compilation mechanism. Our modification to the ACT-R architecture allows us, instead, to specify a prototypical case for matching a production. This is explained in detail in the following section.

4.4 Flexibility and Generalization in Production Rule Pattern Matching

Perhaps the most significant difficulty in authoring production system models (e.g. expert systems) is specifying the conditions under which productions can apply. Because of the lack of a conventional control structure, it is often difficult for the author to forecast exactly the full range of symbolic conditions uder which an action is applicable. Moreover, in dynamic, approximate and uncertain domains (such as a MOUT simulation), the all-or-none symbolic conditions (i.e. either specify a specific value required or else no restriction on that value) that determine production rules' applicability have significant limitations in capturing the loose range of conditions under which a behavior might be applicable. What is desired is the capability of having a production specify a canonical case in which an action is applicable, then have the production system generalize it to related situations. This is similar to human training in which an instructor demonstrates a particular technique but leaves it to the students to learn by experience the details of its applicability.

A similar restriction on matching chunks of information in declarative memory has long been recognized and remediated with the addition of a partial matching mechanism to memory retrieval. That mechanism allows chunks that only partially match the desired pattern specified by a production retrieval request to qualify for matching. A chunk's activation, which represents in ACT-R the likelihood of a chunk being relevant to a particular situation, is decreased by the amount of mismatch, thereby reducing the probability of retrieving that chunk but not eliminating it altogether. The similarity values used in specifying partial matches between chunk values can be viewed as a high-level equivalent to distributed representations (specifically, to the dot-product between representation vectors) in PDP networks [12]. It seems logical to implement the same mechanism for production rule matching, thereby emphasizing the symmetry between the declarative and procedural parts of architecture by unifying their matching mechanisms. Practically, this would allow pieces of knowledge that were specified and used as declarative instances to seamlessly transition to production rules.

Currently, only production rules whose conditions match perfectly to the current state of various information buffers (goal, memory retrieval, perceptual, etc) qualify to enter the conflict set. Since ACT-R specifies that only one production can fire at a time, the rule with the highest expected utility is selected from the conflict set as the one to fire. The utility of a production rule is learned by a Bayesian mechanism as a function of its past history to reflect the probability and cost of achieving its goal. In a manner similar to partial matching in declarative memory, all rules (subject to types restrictions for tractability reasons) will now be applicable but the new mechanism of production rule matching will scale the utility of a rule by the degree to which its conditions match the current state of the buffers. Specifically, the scale utility (*SU*$_p$) of a rule *p* is specified as:

$$SU_p = U_p + \sum_{conds} MP \cdot Sim_{vd}$$

Figure 7: Scaled Utility Equation

where U_p is the usual utility of the rule, and the penalty term is a product of *MP*, a mismatch scaling constant, and Sim_{vd}, the similarity between the actual value *v* present in a buffer and the desired value *d* specified in the production condition, summed over all production conditions. Similarities are 0 for a perfect match, leading to no change in production utility, and negative for less-than perfect matches, leading to decrement in utility that lowers the probability of the rule being selected with the degree of mismatch. The mismatch penalty *MP* can be seen as a regulating factor, with large values trending toward the usual all-or-none symbolic matching.

While this architectural extension remains to be fully exercised, on limited test cases it succeeds in providing the desired approximate and adaptive quality for production rule matching. All things being equal, productions will generalize equally around their ideal applicability condition. However, productions with higher utility will have a broader range of applicability, up to the point where they reach their limits and failures lower their utility, thereby providing a learning mechanism for the range of applicability of production rules. Moreover, the range of applicability of a production rule will be a function of the presence of production rules with similar competing conditions. In the initial learning of a new domain, a few production rules will be generalized broadly as all-purpose heuristics. As more knowledge of the domain is accumulated and new production rules created, the range of application of those rules will be increasingly restricted.

Using this modification to the ACT-R production-matching scheme, no 'hard' boundaries exist between conditions for matching productions, and the boundaries are instead continuous. For example, if production A is appropriate when a doorway is to the front, while production B is appropriate when a doorway is to the left side, both productions may match when a doorway is both ahead and to the left. While specifying directions such as 'left' as a range makes it possible to match a production in a symbolic system to a range of situations, specifying 'left' as a precise direction and allowing productions to match based on similarity to that condition allows both cleaner specification of the underlying representation (i.e., 'left' is 90 degrees to the left instead of between 45 degrees and 135 degrees to the left), and easier authoring of the productions with a reduction in unwanted interactions between pieces of procedural knowledge. In this case, if the author later decided a new production, production C, was appropriate when a doorway was ahead and to the left, adding the new production C to the system would result in that production predominating over the others without any revision of productions A and B.

5. Communication

Planning, as presented above in section 4.1, requires at the least the ability for agents to signal each other. We have provided a grammar that the agents use to communicate that includes signaling, acknowledgment, sending and receiving orders, communication of intention, and specification of the type and location of a contact (e.g., friendly fire, from location (x,y,z)).

The most fundamental of these, simple communication, is potentially non-verbal and simply involves the passing of signals and the acknowledgment of their receipt. For example, saying "On the count of three, go" requires the receipt of the signal "three" while ignoring other signals. In the UT environment, this is implemented by passing text messages between the agents. Although the agents could have passed tokens in a coded language, we chose to use actual English phrases for readability and extensibility to interactions with human players.

The passing of orders typically involves an instruction to execute a schematic plan. These plans include actions such as clearing an L-shaped hallway, supplying covering fire, moving to a particular location, standing guard, providing assistance in storming a particular room, providing covering fire, or retreating from overwhelming fire. Importantly, for this representational shorthand to work, both the order sender and receiver must share these plans. Like real combatants, the agents must know exactly what to expect from each other to function effectively as a unit. These schematic plans often depend on stereotypic doctrinally defined simple communications. For example, when storming a room, attackers typically "stack" outside the doorway. Attackers in front are signaled by the attackers behind that they are in position to enter the room, obviating the need to turn away from a potentially hazardous entrance at a critical moment. Although it is possible for real combatants to signal each other non-verbally (perhaps in this case with a touch on the back), agents in this environment simulate non-verbal messages through the passing of text messages.

In addition to orders, agents can also share information. The most common information shared is a spot report of enemy activity. A spot report includes a brief description of the enemy forces spotted including their numbers and armament if known, their location, and their movement (if any). Other agents may use this information to provide coordinated ambushes and attacks.

6. Action vs. Reaction

The schematic plans outlined above indicate the system is capable of goal directed activity. However, in a real-time system such as this, the environment may change in a way that is incompatible with the current goal. As an example, an agent may have a goal to move towards the doorway of an unexplored room. If an enemy enters the hallway within sight, the agent clearly should abandon exploration and deal with the threat. ACT-R 5.0 provides a mechanism built into the architecture that allows for interruption by critical events – multiple buffers. In this case, a buffer is used to keep track of any perceived threats. Exploratory behavior continues in the absence of a threat, but once a threat is perceived, the perception of the threat interrupts the current system goal and forces the agent to deal with the threat (though the agent could then choose to ignore the threat, that choice still must be made explicitly).

Similarly, occasionally it is desirable to simultaneously pursue two goals, and in effect, "kill two birds with one stone". For example, while moving from one location to another, an informational message may be received. Although it would be simple to abandon the movement to handle the incoming transmission, this is clearly not plausible or desirable. The preferred solution is to continue the agent's movement while handling the transmission. This is also accomplished through the use of a buffer for keeping track of an initiated movement. The human behavioral equivalent is driving from place to place – often once the drive is initiated, other goal-directed behavior can occur without interrupting the drive. It is not that the two goals are being serviced at the same time but that the pursuit of compatible simultaneous goals can be achieved without simultaneous actions – they can be interleaved through the use of an architecture like this.

7. Conclusions

Our use of perceptually plausible spatial representations has two primary benefits: 1) code is much easier to read when written in terms that are readily understood by non-programmers, and 2) human behavior is easier to encode if it can be written in terms that humans would use to describe their behavior. Typically, authoring not only becomes qualitatively simpler, but also is also quantitatively less demanding since symmetrical cases do not need to be re-authored.

The utility of using these spatial functions is in the ease with which they can help structure behavior and write rules. Temporal and spatial aspects of a task often structure real behavior. One continues doing what one just was doing, and continues performing that action where one was just doing it. For example, if a pair of marines is about to storm a room from the stacked position, a pseudo-rule for entering the room could be written as:

> If the goal is to storm a room and a door is directly ahead and near then storm the doorway.

There is no need to carry a pointer around to the doorway about to be entered. It is the one right in front of the marine. No real marine would, once stacked, suddenly head off to a different door. The previous action -- the temporal component -- and the configuration of things around the marine -- the spatial component -- fully determine what is to come next. As a result of this type of coding, exhibited behavior of the cognitive models is (appropriately) highly dependent on the context.

A further refinement to the system we are working on is the use of the ACT-R partial matching facility for spatial processing. For example, it is arguably more plausible to think of "ahead" as having a best interpretation and less-good interpretations that deviate from the prototype. This allows for very flexible matching that shoots for the best match but will accept a decent though imperfect match if no better is available, while also sometimes allowing an imperfect match to be selected over the better alternative.

This refinement was a result of working with the more cumbersome method employed by traditional production systems, in which a symbolic match is a binary condition: either the conditions match or they do not. To get flexible behavior as required in a domain such as this from a strictly symbolic pattern matching system, the conditions must be flexible. Based on our initial experiences, the authoring of flexible conditions is difficult at best and requires careful attention to the potential interactions among conditions. It is quite possible to author a system that encounters a situation where no condition applies exactly, resulting in no match whatsoever despite several near misses. This is one of the major reasons many researchers have characterized systems that apply strict symbolic pattern matching to complex real-world situations as ‘brittle’. The alternative we are developing here is instead to employ flexible matching to inflexible conditions. This greatly simplifies the authoring of productions as well as their interactions.

Teamwork occurs in the current system both formally and informally. An example of informal teamwork is an agent sidestepping behind another friendly agent to get a clear shot at an enemy. Formal teamwork involves schematic plans in which two agents each assume a role and work together according to doctrinal techniques. By having knowledge about doctrine common between agents, intricate and flexible teamwork is possible with minimal communication.

Communication was still, however, a necessary prerequisite to teamwork. Our early efforts involved attempting to infer the goals of a teammate based on their apparent actions. Too often this is a noisy inference, and simple communication easily remedies it. Among human combatants the sometimes-apparent lack of communication turns out to be well-practiced subtle non-verbal communication; these combatants by necessity give each other signals that would not be easily perceived or understood by onlookers. Their common knowledge comes both from doctrine, and from long hours of training together.

Achieving the proper balance between goal-directed and reactive behavior is a challenge. In many cases the only recourse is to view the behavior produced by the agents and determine with the assistance of a subject matter expert whether that behavior is plausible. Often the subject matter experts identify many behaviors as plausible but qualify them as more or less likely. ACT-R provides a useful solution to this dilemma by providing for stochasticity in action selection. This allows a wide range of behavior to be exhibited by cognitive models developed within the ACT-R framework.

Acknowledgements

This work was funded by grant N00014-02-1-0020 from the Office of Naval Research. We thank Eric Biefield for his work on developing code for spatial transformations.

References

[1] Agre, P., & Chapman, D. Pengi: An implementation of a theory of activity. In *Proceedings of the sixth National Conference of the American Association for Artificial Intelligence (AAAI-87)*, Morgan Kaufmann, San Mateo, CA, 1987.

[2] Anderson, J. R., and Lebiere, Christian: *The Atomic Components of Thought*, Erlbaum, Mahwah, NJ 1998

[3] Beetz, M.: Plan-Based Control of Robotic Agents, Lecture Notes in Artificial Intelligence 2554, 2002.

[4] Best, B. J., Scarpinatto, K. C., and Lebiere, C.: Modeling Synthetic Opponents in MOUT Training Simulations Using the ACT-R Cognitive Architecture. In *Proceedings of the 11th Conference*

on Computer Generated Forces and Behavior Representation. University of Central Florida, 2002

[5] Clementini, E., Felice, P. D., and Hernandez, D. Qualitative representation of positional information. Artificial Intelligence, 95:317-356, 1997.

[6] Cutting, J. E. Reconceiving perceptual space. In Perceiving pictures: An interdisciplinary approach to pictorial space, H. Hecht, M. Atherton, & R. Schwartz, Eds. MIT Press, in press.

[7] Frank, A.: Spatial Communication with Maps: Defining the Correctness of Maps Using a Multi-Agent Simulation. *Spatial Cognition II* pp. 80-99, 2000.

[8] Gillis, P. D.: *Cognitive Behaviors for Computer Generated Command Entities*. U.S. Army Research Institute Technical Report, 2000

[9] Klatzky, Roberta L.: Allocentric and Egocentric Spatial Representations: Definitions, Distinctions, and Interconnection, *Spatial Cognition* pp. 1-18, 1998

[10] Loomis, J. M., & Knapp, J. M. Visual perception of egocentric distance in real and virtual environments. In *Virtual and Adaptive Environments*, L. J. Hettinger and M. W. Haas, Eds. Erlbaum, Hillsdale, NJ, in press.

[11] Marine Corps Warfighting Publication (MCWP) 3-35.3, *Military Operations on Urbanized Terrain (MOUT)*

[12] Rumelhart, D.E., & McLelland, J. L. A general framework for parallel distributed processing. In D. E. Rumelhart & J. L. Mclelland, Eds., Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol 1, 1986.

[13] Schank, R.C., & Abelson, R.P. Scripts, plans, goals and understanding. Hillsdale: Erlbaum, 1977.

[14] Siegler, R. S. & Opfer, J. E. The development of numerical estimation: Evidence for multiple representations of numerical quantity. Psychological Science, in press.

[15] Silverman, B. G., Might, R., Dubois, R., Shin, H., Johns, M., & Weaver, R.: Toward a human behavior models anthology for synthetic agent development. In *Proceedings of the 10th Conference on Computer Generated Forces and Behavior Representation*. Norfolk, VA, 2001

[16] Tambe, M., Johnson, W.L., Jones, R. M., Koss, F., Laird, J. E., Rosenbloom, P. S., & Schwamb, K. Intelligent agents for interactive simulation environments. *AI Magazine*, 16, 15-40, 1995.

Author Biographies

BRAD BEST is completing his Ph.D. in Psychology at Carnegie Mellon University where his graduate career has focused on simulations of human problem solving in spatial tasks. He received his B.S. in Computer Science from the University of Detroit and his M.S. in Computer Science from Central Michigan University. In his graduate work there he concentrated on connectionist models of visual systems. In addition to academic pursuits, he has spent several years in industry applying rule-based artificial intelligence to real-world problems. His main research interest is the development of artificial intelligence planning systems with spatial components and their application to complex tasks including navigation, game playing and problem solving.

CHRISTIAN LEBIERE is a Principal Research Scientist at Micro Analysis and Design, Inc. He received his B.S. in Computer Science from the University of Liege (Belgium) and his M.S. and Ph.D. from the School of Computer Science at Carnegie Mellon University. During his graduate career, he worked on the development of connectionist models, including the Cascade-Correlation neural network learning algorithm that has been used in hundreds of scientific, technical and commercial applications. Since 1990, he has worked on the development of the ACT-R hybrid cognitive architecture and is co-author with John R. Anderson of the 1998 book “The Atomic Components of Thought”. His main research interest is cognitive architectures and their applications to psychology, artificial intelligence, human-computer interaction, decision-making, game theory, and computer-generated forces.