UBGCCS 93: Proceedings of the Eighth Annual University at Buffalo Graduate Conference on Computer Science

Daniel F. Boyd (ed.)

Department of Computer Science
State University of New York at Buffalo
226 Bell Hall
Buffalo, NY 14260

Air Battle Simulation: An Agent with "conscious" and "unconscious" layers

Henry Hexmoor
Guido Caicedo
Frank Bidwell
Stuart C. Shapiro
Department of Computer Science
State University of New York at Buffalo
Buffalo, NY 14260
hexmoor@cs.buffalo.edu

Abstract

Intelligent behavior can be generated by reasoning, i.e., "conscious" processes, or by repeating a learned but mindless, successful pattern of interaction with the world, i.e., "unconscious" processes. We have designed and implemented a program, Air Battle Simulation (ABS), that demonstrates "conscious" and "unconscious" layers of an architecture for an agent. We demonstrate how in a dynamic environment, the agent mostly improvises. The agent's reasoned out (i.e., "conscious" actions) are cached in its "unconscious" layer. This agent also learns to improve its "unconscious" processes and learns new actions.

1 Behavior Generation

We are interested in modeling behavior generation by agents that function in dynamic environments. We make the following assumptions for the agent.

- The environment demands continual and rapid acting, e.g., playing a video-game.
- The impact of the agent's action for the agent depends on the situations under which actions are applied and on other agents' actions.
- Other agents' actions are nondeterministic.
- The agent does not know about long term consequences, i.e., beyond the current situation, of its actions.
- The agent is computationally, cognitively resource bound. We assume that the agent needs time to think about best action and in general there is not enough time.

To cope in dynamic environments, an agent which is resource bound needs to rely on different types of behaviors, for instance, reflexive, reactive, situated, and deliberative behaviors. Reflexive and reactive behaviors are predominantly "unconscious" behaviors, situated action may be either "unconscious" or "conscious", and deliberative actions are predominantly "conscious" behaviors. We assume that in general "conscious" behavior generation takes more time than "unconscious" behavior generation.

Reflexive behavior[1] occurs when sensed data produces a response, with little or no processing of the data. A reflex is immediate. The agent has no expectations about the outcome of its reflex. The reflexive response is not generated based on a history of prior events or projections of changing events, e.g., a gradual temperature rise. Instead, reflexive responses are generated based on spontaneous changes in the environment of the agent. These spontaneous changes in one way or another affect the agent, either *overtly* as in "fire near skin" or *covertly* as in peripheral visual perception. In anthropomorphic terms, this is innate behavior that serves directly to protect the organism from damage in situations where there is no time for conscious thought and decision making, e.g., the withdrawal reflex when sticking one's hand into a fire.

Reflexive behavior does not require conscious reasoning or detailed sensory processing, so our lowest level, the Sensori-Actuator Level, is charged with producing these behaviors. Our initial mechanism for modeling reflexive behavior is to design rules of the form $T \rightarrowtail A$, where T is a trigger and A is an action. A trigger can be a simple temporal-thresholding gate. The action A is limited to what can be expressed at the Sensori-Actuator Level, and is simple and fast.

Reactive behavior requires some processing of data and results in ***situated action*** [Suchman, 1988]. However, its generation is unconscious. ***Situated action*** refers to an action that is appropriate in the environment of the agent. In anthropomorphic terms, this is a learned behavior. An example would be *gripping harder when one feels an object is slipping from one's fingers*, or *driving a car and tracking the road.* We use the term *tracking* to refer to an action that requires continual adjustments, like steering while driving. Examples of this type of reactive behavior are given in [Payton, 1986, Anderson et al., 1991].

Situated behavior requires assessment of the state the system finds itself in (in some state space) and acting on the basis of that. In anthropomorphic terms, this is learned behavior that requires only shallow reasoning. It might be modeled by the workings of a finite state automaton, for example, the Micronesian behavior described in [Suchman, 1988]. Situated action is used in *reactive planning* [Agre & Chapman, 1987, Firby, 1987, Schoppers, 1987]. Reactive planners and systems of situated behaviors share the following properties:

- Applicability of only one action is decided upon at any one time.
- Applicability of an action is decided based on the current situation and not on the history of what has happened.
- No predictions are made about what will be true after completing an action.

Deliberative behavior requires considerable processing of data and reasoning which results in action. In anthropomorphic terms, this is learned behavior that requires deep reasoning that can be modeled by a Turing machine (or first order logic), for example explicit planning and action.

We have developed an architecture, Grounded Layered Architecture with Integrated Reasoning, GLAIR [Hexmoor et al., 1992]. GLAIR provides a natural framework for modeling distinct types of behavior generation. GLAIR specifies three layers; named knowledge, perceptuo-motor and sensory-actuator. The knowledge layer is considered to contain conscious processes of the agent and in our current implementations we are using SNePS for encoding knowledge at this layer. The other two layers are considered to contain unconscious processes.

1.1 PMA

At the perceptuo-motor level, the behaviors resulting in physical actions are generated by an automaton, which we will call a PM-automaton (PMA) [Hexmoor & Nute, 1992]. In other words, PMA are representation mechanisms for generating behaviors at an "unconscious" level. PMA is a family of finite state machines represented by <Rewards, Goal-Transitions, Goals, Action-Transitions, Actions, Sensations>. Goals, Rewards, and Goal-Transitions in a PMA can be empty. The simplest

[1] E.g., visual reflexes in [Regan & Beverly, 1978]: Here responses are generated to certain visual stimuli that do not require detailed spatial analysis.

class of PMA is when all three of these elements are empty. Below is the list of combinations of tuples, each defining a class of PMA.

- < NIL, NIL, NIL, Action-Transitions, Actions, Sensations>
- <NIL, Goal-Transitions, Goals, Action-Transitions, Actions, Sensations>
- <Rewards, NIL, NIL, Action-Transitions, Actions, Sensations>
- <Rewards, Goal-Transitions, Goals, Action-Transitions, Actions, Sensations>

Actions is a set of primitive actions. Sensations are a set of atomic (i.e., nondecomposable) percept types in a PMA. For example, a percept type for a mobile robot that walks in building hallways can be *its distance to the wall.* In Air Battle Simulation (ABS), *relative distances from the enemy in X, Y , and Z axes* are percepts. We assume that all possible situations for a PMA can be specified by one or many Sensations, i.e., percept types. We call a pattern of Sensation types that are used for input to a PMA a Situation. In ABS, <*distanceX, distanceY, distanceZ, orientation*> is a Situation. Henceforth, we will refer to values of a situation input to a PMA as Situation instances. In ABS, <*close-front, close-right, close-above, parallel*> is a situation instance. Goals are descriptions of desirable conditions for a PMA. In PMA classes with PMA, Goals are used to partition a PMA. Rewards are static goodness values associated with situations. These are determined apriori to PMA execution and remain unchanged throughout.

Action-Transitions (AT) are transformations that generate behaviors for execution, (see Figure 1). The simplest ATs are mappings of Sensations $\mapsto$ Action. Below is a list of AT types. ATs can be disabled by inhibiting them. This is useful when the agent wants to override unconscious behaviors generated by PMA with conscious behaviors generated by the knowledge layer.

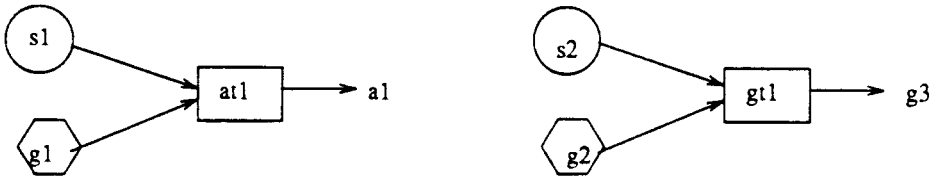

Figure 1: An action transition is shown on the left and a goal transition is shown in the right.

- Sensations $\rightarrowtail$ Action
- Previous action $\times$ Sensations $\rightarrowtail$ Action
- Goal $\times$ Sensations $\rightarrowtail$ Action
- Tendency $\times$ Sensations $\rightarrowtail$ Action
- Previous action $\times$ Goal $\times$ Sensations $\rightarrowtail$ Action
- Previous action $\times$ Goal $\times$ Tendency $\times$ Sensations $\rightarrowtail$ Action
- Tendency $\times$ Goal $\times$ Sensations $\rightarrowtail$ Action

ATs may consider goals as antecedents. ATs may consider previous actions in deciding what to do next. We may take into account an estimated accumulated reward for actions. We call the latter, tendencies. Tendencies are computed using reinforcement based learning techniques [Sutton, 1988].

Goal-Transitions (GT) are transformations that update goals when the current goal is satisfied. GT is Goal1 × Sensations ↦ Goal2 where Goal1 and Goal2 are goals.

The PMA maintains the current goal. When the latest sensations along with the current goal match an action transition, that action transition activates an action which is then executed.

The primary mode of acquiring a PMA is by converting plans in the knowledge level into a PMA by a process described in [Hexmoor & Nute, 1992]. Later in this paper we describe an stepwise learning scheme for acquiring a PMA.

2 Air Battle Simulation

We have written a program, Air Battle Simulation (ABS), that simulates world war I style airplane dog-fights. ABS is an interactive video-game where a human player plays against a computer driven agent. The game runs on SparcStations and starts up by displaying a game window, (see Figure 2), and a control panel window, (see Figure 3). The human player's plane is always displayed in the center of the screen. The aerial two-dimensional position of the enemy plane is displayed on the screen with the direction of flight relative to the human player's plane. The human player looks at the game screen to determine his airplane's position and orientation with respect to the enemy's plane. (S)he then uses the control panel to choose a move. A move is a combination of changing altitude, speed, and direction. When the human player presses the go button, the computer agent also selects a move. The game simulator then considers the human player's move and the computer agent's move to determine the outcome of moves, and updates the screen and the accumulated damage to planes. ABS simulates simultaneous moves this way. If a player's plane is close in altitude and position to the enemy plane, and the enemy is in frontal sight, the latter is fired on automatically (i.e., firing is not a separate action). The levels of damage are recorded in a side panel, and the game ends when one or both of the two player's planes are destroyed. The computer agent is developed in accordance with the principles of the GLAIR architecture. Figure 4 schematically represents the structure of the computer agent. Initially, the agent has not acquired a PMA, the computer agent uses conscious level reasoning, i.e., SNePS rules, to decided what move to make. Once transitions are learned and cached in a PMA, the computer agent uses the PMA for deciding its next move whenever possible. By adding learning strategies, a PMA can be developed that caches moves decided at the knowledge level for future use. Here again, learning can be used to mark PMA moves that prove unwise and to reinforce moves that turn out to be successful. We are exploring these learning issues. The computer agent particularly demonstrates real time behaviors and the inter-level alignment mechanism.

3 A Stepwise Learning Scheme

In ABS, when the knowledge layer determines an action, it submits a tuple of action A and a goal G to PM-level. If the goal is different from the current goal in PMA, a new goal transition GT is learned. Let the current situation in PMA be S. GT is then S × current goal ↦ G. If there is no action transition corresponding to S, G, and A, a new action transition is learned. AT is then S × current goal ⊢ A. Finally, if the goal is different from the current goal in PMA, the current goal is updated.

We started ABS with an empty PMA and as the game was played, transitions of the PMA were learned. Figure 5 shows typical ABS transitions learned. Also as the transitions were learned, when similar situations occurred, if there was an appropriate PMA response, the PMA executed that action. As the game was played, we observed that the agent became more reactive since the PMA was used to generate behaviors instead of the knowledge layer.

Figure 5 shows a small sample of learned PMA transitions while playing ABS.

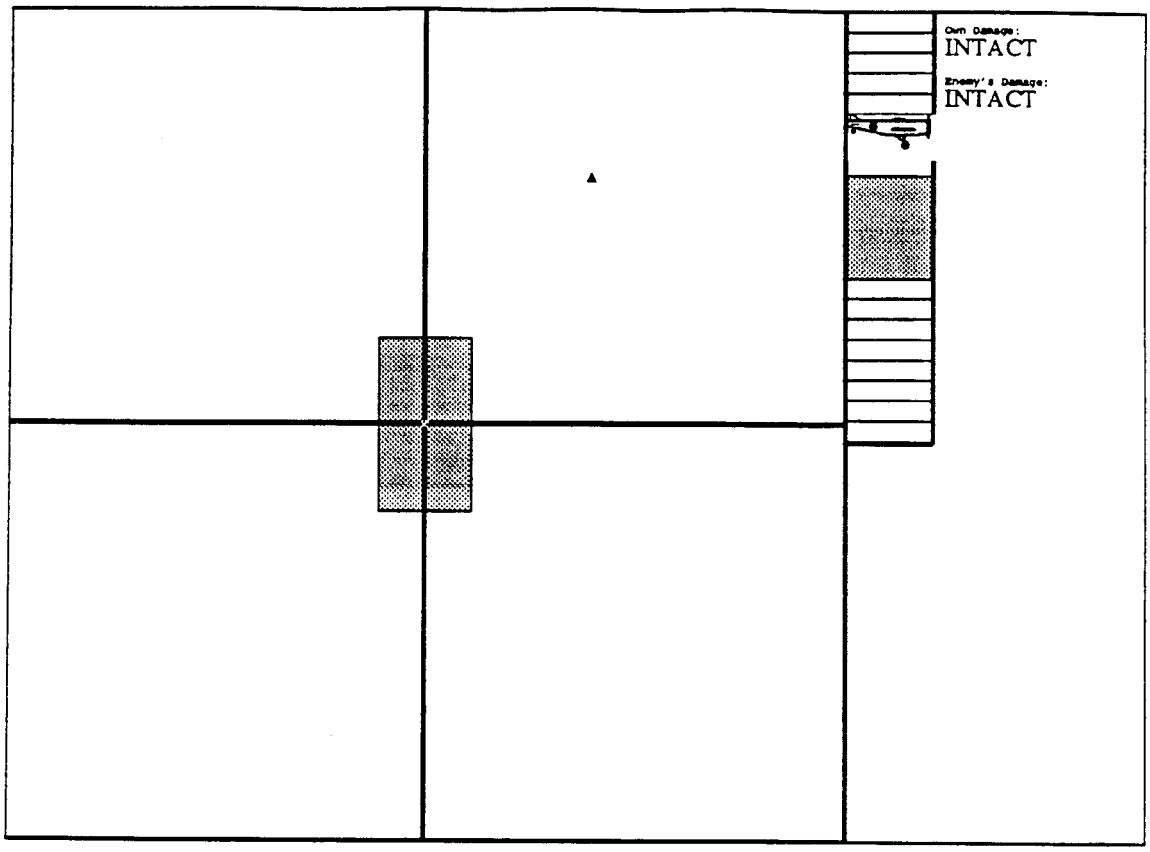

Figure 2: Air Battle Simulation game window. The enemy is shown by a small triangle in the upper right quadrent. This figure shows the enemy fleeing, flying parallel, and at a higher relative altitude. The shaded regions denote shooting range. If the enemy is in close relative altitude and distance (i.e., appears in both shaded regions) and in front (i.e., upper quadrants), it will be under fire.

4 Improving "Unconscious" Behaviors

The rules of a PMA are pairs of situation/action. As it turns out, a situation can be paired up with multiple actions. The object of learning here is to learn which actions when associated with a situation yield a better result, i.e., the pilot ends up in a more desirable situation.

Some situations in ABS are more desirable for the pilot than others, e.g., being right behind the enemy and in shooting range. Let's assume that we can assign a goodness value to each situation s between -1 and 1, G(s). As the pilot makes a move, it finds itself in a new situation. This new situation is not known to the pilot since it also depends on the other pilot's move. Since the new situation is not uniquely determined by the pilot's move, the pilot's view of the game is not markovian.

Q(s,a) is the evaluation of how appropriate action a is in situation s. R(s,a) is the goodness value of the state that the pilot finds itself after performing a in situation s. R(s,a) is determined as the game is played and cannot determined beforehand. This is called the immediate reward. γ is a parameter that we plan to vary that to determine how important it is to be in the state that the pilot ends up in after his move. In reinforcement based learning this is known as the discount factor.

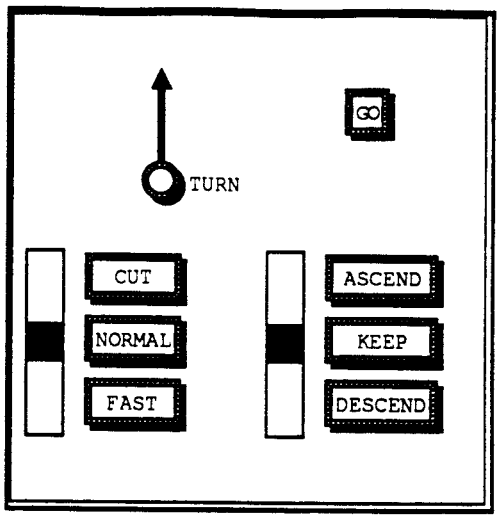

Figure 3: Air Battle Instrument Panel window. To select/change a move, the human player pushes one of the buttons in each column and toggle on the TURN dial. CUT, NORMAL, FAST are used for speed change. CUT simulates cutting the engine, NORMAL simulates maintaining a cruising speed, and FAST simulates a constant high speed. ASCEND, KEEP, DESCEND are used for altitude change. ASCEND simulates gaining elevation, KEEP simulates maintaining elevation, and DESCEND simulates losing elevation. Pushing the GO button submits the move to the game simulator.

$Q(s,a) = R(s,a) + \gamma \max_k Q(s',k)$ where situation s' results after the pilot performs a in s. γ is between 0 and 1.

At the start of game, all $Q(s,a)$ in PMA are set to 1. As the game is played, Q is updated. As of this writing we are experimenting with setting parameters for Q.

5 Observing Successful Patterns of Interaction in the World

We assumed that the agent does not know about long term consequences of its actions. Furthermore, the reinforcement based learning we described in the previous section assumes a markovian environment. That is, the agent believes the world changes only due to its own actions. This makes it necessary to observe interactions with the world in order to learn sequences of actions. Over a finite number of actions, when the agent observes a substantially improved situation, chances are he has found a successful *Routine*. We record such detected *Routines* and as they reoccur, we increase our confidence in them. When our confidence in a *Routine* reaches a certain level, a concept is created at the knowledge layer of GLAIR for the routine and from then on, this routine can be treated as a single action at the knowledge layer.

6 Summary and Future work

We have implemented a video-game system that exhibits "conscious" and "unconscious" behaviors. This system becomes faster as it learns "conscious" behaviors. It improves its "unconscious' rules by reinforcement based learning. It learns complex actions as *Routines*, i.e., a set of recurring actions, at the knowledge layer.

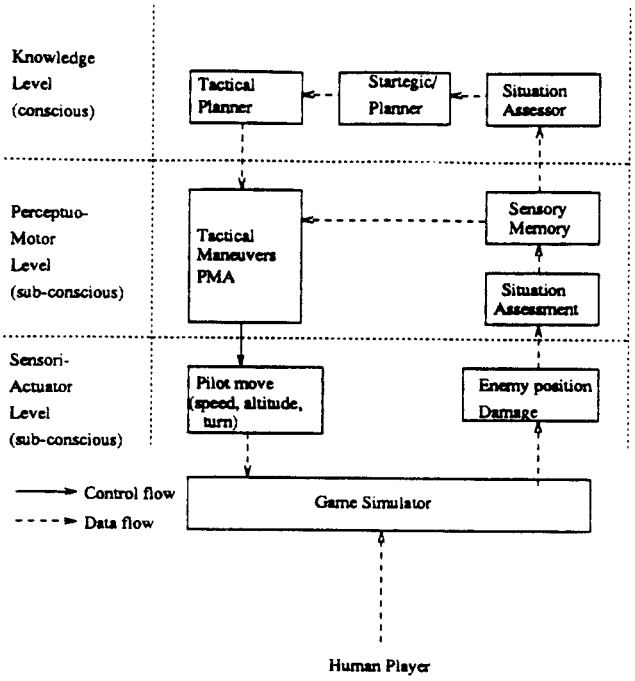

Figure 4: Schematic representation of the Air Battle Simulation GLAIR-agent.

We plan to explore other learning techniques such as experimentation as a form of learning [Shen. 1989]. We are also interested in developing experiments that will help in psychological validation of GLAIR.

References

[Agre & Chapman, 1987] Agre, P. E. & Chapman, D. (1987). Pengi: An implementation of a theory of activity. In *Proceedings of AAAI-87, Seattle, Wa.* (pp. 268–272).

[Anderson et al., 1991] Anderson, S., Hart, D., & Cohen, P. (1991). Two ways to act. In *ACM SIGART Bulletin* (pp. 20–24). ACM publications.

[Firby, 1987] Firby, R. J. (1987). An investigation into reactive planning in complex domains. In *Proceedings of AAAI-87* (pp. 202–206).

[Hexmoor et al., 1992] Hexmoor, H., Lammens, J., & Shapiro, S. (1992). *An Autonomous Agent Architecture for Integrating Perception and Acting with Grounded, Embodied Symbolic Reasoning.* Technical Report CS-92-21, Dept. of Computer Science, SUNY at Buffalo, New York.

[Hexmoor & Nute, 1992] Hexmoor, H. & Nute, D. (1992). *Methods for Deciding what to do next and Learning.* Technical Report AI-1992-01, AI Programs, The University of Georgia, Athens, Georgia. Also available from SUNY at Buffalo, CS Department TR-92-23.

[Payton, 1986] Payton, D. (1986). An architecture for reflexive autonomous vehicle control. In *Proceedings of Robotics Automation* (pp. 1838–1845).: IEEE.

[Regan & Beverly, 1978] Regan, D. & Beverly, K. (1978). Looming detectors in the human visual pathways. In *Vision Research 18* (pp. 209–212).

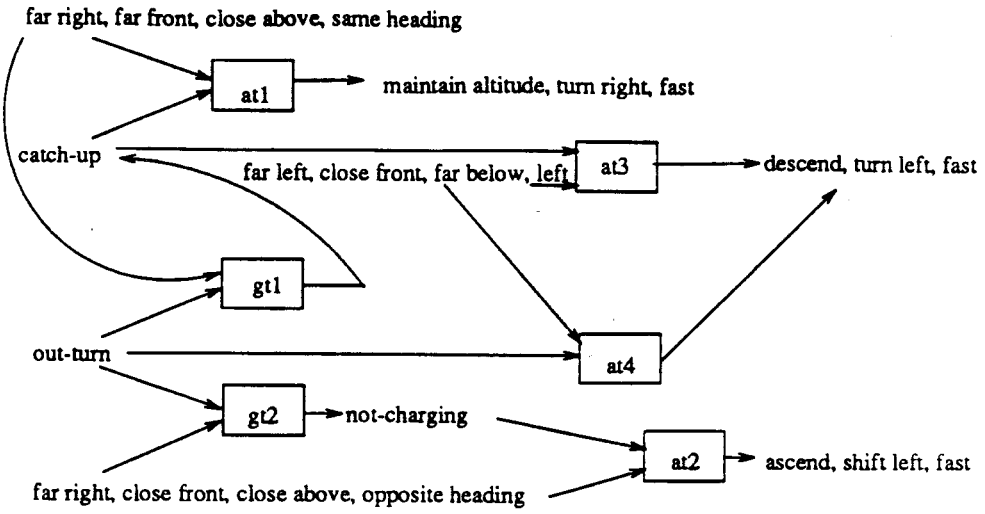

Figure 5: Sample of Learned Action and Goal Transitions

[Schoppers, 1987] Schoppers, M. J. (1987). Universal plans for unpredictable environments. In *Proceedings 10th IJCAI* (pp. 1039–1046).

[Shen, 1989] Shen, W.-M. (1989). *Learning from the Environment Based on Actions and Percepts.* PhD thesis, Carnegie Mellon University.

[Suchman, 1988] Suchman, L. A. (1988). *Plans and Situated Actions: The Problem of Human Machine Communication.* Cambridge University Press.

[Sutton, 1988] Sutton, R. (1988). Learning to predict by the methods of temporal differences. In *Machine Learning 3* (pp. 3–44).