Animation through Reactions, Transition Nets and Plans

Bonnie Webber Norman Badler
Center for Human Modeling and Simulation
Department of Computer & Information Science
University of Pennsylvania*

Abstract

We describe a framework for creating animated simulations of virtual human agents. The framework allows us to capture flexible patterns of activity, as well as reactivity to a changing environment. Both lead to variation in how an animated simulation will be realized. In addition, because different parts of an activity make different demands on an agent's resources and decision-making, our framework allows special-purpose reasoners and planners to be associated with only those phases of an activity where they are needed.

Introduction

Conventional animations often seek to re-create "life" through the artistic skills of an animator who transforms his or her observations, experience, and intuition into the images, shapes, and movements that make for believable characters (Thom81). Recent automated techniques for animation aim to ease the animator's burden. For example, dynamics can be used to animate particles or objects (Hahn88; Wilh90), and "flocking" can be used to constrain interactions between figures (Reyn87; Tu84).

Producing animated people, however, seems to require more than the existing physical or manual toolset. One response to this difficulty is "performance animation", where live actors perform, while sensing systems monitor various body landmarks labeled by markers or electromagnetic sensors (Badl93a; Robe94). While this provides motion data of unquestioned realism, it is only one instance of a performance: different conditions can lead to different motions because people adapt their behavior to circumstances through their unconscious reactions and conscious decision-making,

What we note here is that *synthetic humans must also be able to adapt to circumstances if we want them to share human qualities.* Sensed human motions are not sufficient because we do not know how to readily adapt them to other circumstances. Even when people's behavior essentially follows a pattern, as in going to a supermarket, playing a game, or troubleshooting and/or repairing equipment, they sense the world around them, react and make choices in ways that adapt the pattern to circumstances. It is supporting flexible patterns of activity that motivates much of our research. We see animation as an integration of a rich collection of interacting techniques, organized in a principled, structured representation (Badl93b). Here we show its use in representing flexible patterns of human activity.

We start by briefly reviewing our two-level architecture for intelligent agents, then comment briefly on previous computational approaches to patterns of human activity, and then show how PaT-Nets capture such patterns for the production of animated simulations.

The Agent Architecture

An *agent* is an object that is able to take action by virtue of having a sense-control-action (SCA) loop to produce locally-adaptive (reactive) behavior. In general, behaviors are considered "low level" capabilities of an agent, such as being able to locomote [to], reach [for], look [at], *etc.*

An agent also manifests high-level patterns of activity and deliberation, which in turn can affect the immediate formulation and parameters of an SCA loop. Such patterns are captured in our framework through parallel state-machines we call "Parallel Transition Networks", or PaT-Nets. PaT-Nets can sequence actions based on the current state of the environment or of the system itself, and represent the tasks in progress, conditions to be monitored, resources used, and temporal synchronization. An agent's deliberations both

*The demonstration described here, Hide and Seek, was designed and implemented by Roxana Cantarovici, Sonu Chopra, Christopher Geib, Michael Moore, Barry Reich and Thomas Trias, with help from Welton Becket. The authors would like to thank Barry Reich as well for his comments on the paper. This research has been partially supported by DMSO DAAH04-94-G-0402; ARPA DAMD17-94-J-4486; U.S. Air Force DEPTH through Hughes Missile Systems F33615-91-C-0001; Air Force DAAH04-95-1-0151; NSF IRI95-04372; ARO DURIP DAAH04-95-1-0023; and ARPA AASERT DAAH04-94-G-0362.

prior to and during action can be captured through special purpose reasoners and planners associated with specific states of a network.

In this framework, the agent can instantiate PaT-Nets to accomplish goals (*e.g.*, go to the supply depot and pick up a new motor), while low-level control is mediated through direct sensing and action couplings in the SCA loop (*e.g.*, controlling where the agent's feet step and making sure that s/he doesn't run into or trip over any obstacles). By linking numerical feedback streams (SCA loops) and state controllers (PaT-Nets supported by special-purpose reasoners and planners), we believe it is possible to obtain maximum flexibility and maintain appropriate levels of specification in the animated simulation of virtual human agents (Badl93b; Beck93).

The rest of this section briefly describes features of SCA loops and PaT-Nets. For more detail, see (Badl95).

Low-Level Control: SCA Loops

The *behavioral loop* is a continuous stream of floating point numbers from the simulated environment. Simulated sensors map these data to the abstract results of perception and route them through control processes, each of which is independently attempting to solve a minimization problem. The results go to simulated effectors or motor actions that enact changes on the agent or the world. This loop operates continuously.

The behavioral loop is modeled as a network of interacting *sense*, *control*, and *action* (SCA) processes, connected by arcs across which only floating point messages travel. An individual path from sensors to effectors is referred to as a *behavioral net*. It is analogous to a complete behavior in an "emergent behavior" architecture such as Brooks' *subsumption architecture* (Broo86), except that nodes may be shared between behaviors, and arbitration (competition for effector resources) may occur throughout the behavioral path and not just at the end-effector level. The behavioral loop is modeled as a network with floating point connections in order to allow the application of low-level, unsupervised, reinforcement learning in the behavioral design process (Beck95). Here we briefly describe the components of an SCA loop.

Sensory Nodes Sensory nodes model or approximate the abstract, geometric results of object perception. They continuously generate signals describing the polar coordinate position (relative to the agent) of a particular object or of all objects of a certain type within a specified distance and field of view.

Control Nodes Control nodes model the lowest level influences on behavior. For example, control of locomotion is loosely based on Braitenberg's *love* and *hate* behaviors (Brai84), here called *attract* and *avoid*. Control nodes are formulated as explicit minimizations using outputs to drive inputs to a desired value (similar to Wilhelms' (Wilh90) use of Braitenberg's behaviors). They typically receive input signals directly from sensory nodes, and send outputs directly to action nodes, though they could be used in more abstract control situations.

Action Nodes Action nodes connect to the underlying human body model and directly execute routines defined on the model (such as walking, balance, hand position, and torso orientation) and arbitrate among inputs, either by selecting one set of incoming signals or averaging all incoming signals. An example is the *walk controller*, which decides where to place the agent's next footstep and then connects to the locomotion generator (Badl93b; Moor95) to achieve the step.

Our main use of SCA loops to date has been in locomotion reasoning.

High-Level Control: PaT-Nets

PaT-Nets are finite state machines with message passing and semaphore capabilities (Beck94; Douv95). Nodes are associated with processes that can invoke executable behaviors, other PaT-Nets, or specialized reasoners or planners. Invocation occurs when a node is entered. An arc transition between nodes in a PaT-Net may check a local condition evaluated within the PaT-Net or a global condition evaluated in an external environment. Arcs are prioritized, and a transition is made to a new node by selecting the first arc with a true condition. Nodes may also support probabilistic transitions, reflecting the probability of transitioning to another node at a given clock tick. *Monitors* associated with a PaT-Net will execute an action if a general condition evaluates to `true`, regardless of the current state. In addition, a PaT-Net may have local state variables available to all processes and conditions, and may also take parameters on instantiation.

A running network is created by making an instance of the PaT-Net class. All running PaT-Net instances are embedded in a Lisp operating system that time-slices them into the overall simulation. While running, PaT-Nets can spawn new nets, communicate with each other, kill other nets, and/or wait (sleep) until a condition is met. Running nets can, for example, spawn new nets and then wait for them to exit (effectively a subroutine call), or run in parallel with the new net, while maintaining communication with it. Because PaT-Nets are embedded in an object-oriented structure, new nets can be defined that override, blend, or extend the functionality of existing nets.

Patterns of Activity

It has long been recognized that much of everyday human activity falls into patterns. In early work on computer-based story understanding, such patterns were captured in structures called "scripts" or

"schemata" and were used to fill in what hadn't been said in a story as well as what had been said (Scha77).

Scripts/schemata have also be used in generating behavior. For example, in "hierarchical planning" (Sace77; Wilk88), a plan operator specifies a partially ordered pattern of actions or sub-goals that can be used to achieve a particular goal.

Here we want to make three points about patterns of activity:

(1) Patterns have different sources: they may be idiosyncratic, peculiar to an individual; they may be cultural, simplifying interactions between people (Cass94); they may be occupational, as in a medic's initial assessment of a trauma patient (Chi95); or they may be recreational, as in the pattern of a game.

(2) Patterns vary in their rigidity. They range from low-level fixed patterns one may do daily without thinking – for example, putting one's left shoe on first and then one's right – to high-level patterns of engagement where the particulars are subject to variation such as in having breakfast, making dinner, going out for dinner, playing golf, *etc.* The variation may reflect low-level reactions to circumstances or high-level decisions.

(3) People may be engaged simultaneously in multiple patterns of activity. Their resulting behavior may therefore reflect their allocating different resources to different patterns, using one pattern in support of another, time-slicing patterns or blending patterns.

The next section shows how PaT-Nets can be used to capture patterns of activity in the game of Hide and Seek. (A video demonstrating the results accompanies this paper.)

Simulating Hide and Seek through PaT-Nets

While there are many different sets of rules for Hide and Seek, most involve one player (who is "it" or the "seeker") first averting his/her eyes while the other players hide, then setting out to find at least one of them, then engaging in some competition with whomever is found, which may then lead to that other player becoming "it" in the next round. Thus Hide and Seek manifests a pattern of behavior whose realization may vary from instance to instance as players hide in different places, as different players are found, and as the competition between the player who is "it" and the player that is found yields different results. Here we show how this kind of pattern is easily supported in PaT-Nets.

The high-level controller, or *PlayNet*, for simulating one common version of Hide and Seek is illustrated in Figure 1. In this version, a hider, once hidden, does not move until seen by the seeker, at which time the hider attempts to run home without being tagged by the seeker. In Figure 1, the node labelled **Sync** causes players to wait until all the players are "home", at which point the seeker may begin counting (**Count**) and the other players start to hide (**Hide**). The node labelled **Evade** has a player running to home base, while avoiding the seeker. (Currently this sort of multi-goal behavior is hand-coded while a more general approach to multiple goal integration is worked out.) Dashed transitions in the network represent a change in role from hider to seeker or *vice-versa*. The **at home** condition becomes true when the agent is at home-base, while the **safe** condition becomes true when the seeker notices that the hider he is pursuing has made it home.

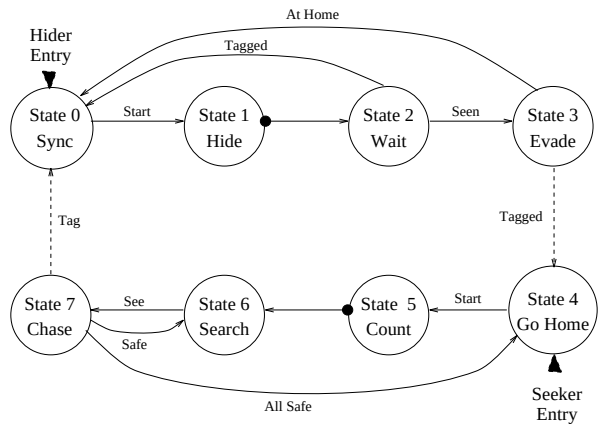

Figure 1: PlayNet for Hide and Seek

While a *PlayNet* specifies an entire game of Hide and Seek, each player has its own copy, which records features of the game specific to that player (*e.g.*, whether the player has been seen by the seeker or made it home safely, *etc.*), as well as features of the global state of the game (*e.g.*, whether all hiders have made it home safely). (We have not yet examined the consequences of having different PaT-Net "rules" for different players.)

To support agent coordination, two types of communicative mechanisms have been used in Hide and Seek. First, the environment itself is used: Players have limited (simulated) perceptual abilities which allow them to see unobstructed nearby things in their environment. This supports the coordination involved in **see/seen** in Figure 1. Second, coordination of all agents is handled through semaphors. For example, in Figure 1, count **done** is broadcast to all agents, as are **tag/tagged** and **at home/safe**, since such information is relevant to all players but cannot be insured known through the players' limited perception.

Focussing now on the processes associated with individual nodes of a *PlayNet*, these may either be intrinsically terminating, as in **count** (count to 10), indicated by a black dot on the arc, or externally terminating by virtue of some event in the environment. For example, **search** is only terminated when circumstances provide the seeker with a hider. Note that intrinsically termi-

nating processes may also be interrupted by external events, stopping them before completion. For example, **evade** (*i.e.*, run home while evading the seeker) is intrinsically terminated by the hider reaching home, but may be interrupted by the hider being tagged. For PaT-Nets to support this, actions must be cleanly preemptable, so that the simulation programmer can pass in a condition for premature termination.

The use of special purpose reasoners and planners in the context of specific processes is illustrated in two of the nodes of Figure 1 – **hide** and **search**. Currently during **hide**, a special purpose reasoner is invoked when the node is entered, that evaluates the set of hiding places the agent knows about and chooses one that maximizes the agent's preference criteria. Each agent has different preferences regarding a hiding place – differing by the importance of its distance from the seeker, the number of available exits, *etc.* Once a hiding place is chosen, the agent moves towards it. A more sophisticated reasoner for choosing hiding places might carry out noticing and evaluating hiding places in parallel, while the player moves away from "home", to avoid wasting time by thinking without moving.

The use of special purpose planners is illustrated in the **search** node. Searching is viewed as choosing a place to look for a hider, and then doing what one needs to get there. The planners that support these two activities are described in more detail in the following sections.

General purpose planning

The function of general purpose planning is to expand high-level goals into a contextually appropriate structure of more basic actions. In Hide and Seek, general purpose planning is used to formulate and maintain a plan for satisfying a seeker's goal of finding a hider. ItPlanS (Geib95) is a hierarchical planner, in which hierarchical expansion of a plan structure to successively lower levels of abstraction only takes place to the degree necessary to determine the next action to be carried out. The incremental nature of the resulting plan allows ItPlanS to make commitments at the appropriate level of detail for action while not committing itself to future actions that might be obviated by changes in the world. ItPlanS is described in more detail in (Geib95).

Search planning

Since human agents have a limited field-of-view, they must search to find objects that are outside of it. We have isolated this reasoning in a specialized module, a *search planner*, that translates information acquisition goals to high-level physical goals to explore parts of the environment (Moor93). To search for an object, an agent must know (or discover during search) the regions of space where the object might be.

Searches terminate successfully when a referent object is "seen" in the environment, and unsuccessfully when there are no more regions to explore or if the environment changes in a way that obviates further search. For example, in Figure 1, the seeker does not continue to **search** after all hiders have returned home, even if parts of the environment remain unexplored.

Our approach to search planning relies on maintaining information about the state of a heuristic search on an internal map. It uses distance from the agent to order regions for exploration. Two lists of regions are maintained by the search planner, an *open* list of regions yet to be explored and a *closed* list of regions which have been explored. On each iteration, the closest region on the open list is selected to be explored, using Pemberton and Korf's Incremental Best-First Search algorithm (Pemb92). ItPlanS then generates a plan for exploring that region, opening any doors necessary along the way. After executing each action in this plan, ItPlanS observes the resulting world to determine if the desired object has been located. New doors and regions observed during the action are added to both the map and the open list.

Conclusion

We have just begun to explore the use of PaT-Nets as high-level controllers of agent activity, but have already identified many questions we want to explore. They include:

- the viability of confining high-level reasoning and planning to particular nodes of a PaT-Net. It may be that in other activities, reasoning and planning need a more global purview or need to be interleaved in as yet unforeseen ways.
- the range of activities worth framing in a net-based approach: the greater the number of nodes, the greater the connectivity between them, and the greater the complexity of the arc conditions, the less a net-based approach seems to make sense. In other words, we would like to identify criteria we can use to decide when to employ a first-principles planner and when we can use a network of pre-determined decision points. It is possible that learning a task or becoming more expert at something entails a migration from planning as a node to more focused networks of choices.
- a clear regimen for mapping instructions for a task or game (including warnings and constraints) to one or more communicating PaT-Nets.

On the other hand, we are now using PaT-Nets in several different projects within the Center, and a graphical interface to PaT-Nets ("Visual Jack") is under development that will enable designers to specify much of a PaT-Net graphically, as in Figure 1. Our experiments so far indicate that the representational efficacy of this architecture of behavioral reaction, transition networks, and symbolic planning is necessary for modeling actions of human-like agents. Whether it is sufficient is a question for the future.

References

Badler, N. I., Hollick, M. J. and Granieri, J. Real-Time Control of a Virtual Human using Minimal Sensors. *Presence* **2(1)** (1993) 82–86.

Badler, N. I., Phillips, C. W. and Webber, B. L. *Simulating Humans: Computer Graphics Animation and Control* (Oxford University Press, New York, 1993).

Badler, N. I., Webber, B. L., Becket, W., Geib, C., Moore, M., Pelachaud, C., Reich, B. and Stone, M. Planning for Animation. To appear in *Computer Animation*, ed. N. Magnenat-Thalmann and D. Thalmann (Prentice-Hall, 1995).

Becket, W. The *Jack* Lisp api. Technical Report MS-CIS-94-01/Graphics Lab 59, University of Pennsylvania, Philadelphia, PA 19104-6389, 1994.

Becket, W. *Reinforcement Learning for Reactive Navigation of Simulated Autonomous Bipeds* (PhD thesis, University of Pennsylvania, 1995).

Becket, W. M. and Badler, N.I. Integrated Behavioral Agent Architecture. in *The Third Conference on Computer Generated Forces and Behavior Representation* (Orlando, FL, 1993).

Braitenberg, V. *Vehicles: Experiments in Synthetic Psychology* (MIT Press, Cambridge, MA, 1984).

Brooks, R. A Robust Layered Control System for a Mobile Robot. *IEEE Journal of Robotics and Automation* (1986) 14–23.

Cassell, J., Pelachaud, C., Badler, N., Steedman, M., Achorn, B., Becket, W., Douville, B., Prevost, S. and Stone, M. (1994). Animated Conversation: Rule-Based Generation of Facial Expression, Gesture and Spoken Intonation for Multiple Conversational Agents. *Computer Graphics*, Annual Conference Series, 413–420.

Chi, D., Webber, B., Clarke, J. and Badler, N. Casualty Modeling for Real-Time Medical Training. Submitted to *Presence*, Special issue on human modelling.

Douville, B. PaT-Net User's Guide. Technical Report. Department of Computer and Information Science, University of Pennsylvania, 1995.

Geib, C. *The Intentional Planning System: ItPlanS* PhD thesis, Dept of Computer & Information Science, University of Pennsylvania, May 1995.

Hahn, J. K. Realistic Animation of Rigid Bodies, *Computer Graphics* **22(4)** (1988) 299–308.

Moore, M. B. Search Plans. (PhD Dissertation Proposal) Technical Report MS-CIS-93-56/LINC LAB 250/IRCS-93-29. Department of Computer and Information Science, University of Pennsylvania, 1993.

Moore, M.B., Geib, C.W. and Reich, B.D. Planning and Terrain Reasoning. *AAAI Spring Symposium on Integrated Planning Applications*, March 1995, Stanford CA. (Also appears as Technical Report MS-CIS-94-63, University of Pennsylvania, 1995).

Pemberton, J.C. and Korf, R.E. Incremental path planning on graphs with cycles. *Proc. First International Conference on AI Planning Systems* Chicago IL, June 1992, pp. 179-188.

Reich, B.D., Ko, H., Becket, W. and Badler, N. Terrain Reasoning for Human Locomotion. *Proceedings of Computer Animation '94* (Geneva, IEEE Computer Society Press, 1994) 996–1005.

Renault, O., Magnenat-Thalmann, N. and Thalmann, D. A Vision-Based Approach to Behavioral Animation. *The Journal of Visualization and Computer Animation* **1(1)** (1990) 18–21.

Reynolds, C.W. Flocks, Herds, and Schools: A Distributed Behavioral Model. *Computer Graphics* **21(4)** (1987) 25–34.

Reynolds, C.W. Not Bumping into Things. *SIGGRAPH Course 27 Notes: Developments in Physically-Based Modeling* (ACM SIGGRAPH, 1988) G1–G13.

Robertson, B. Caught in the Act. *Computer Graphics World* **17(9)** (1994) 23–28.

Sacerdoti, E. *A Structure for Plans and Behavior*, New York: American Elsevier, 1977.

Schank, R. and Abelson, R. *Scripts, Plans, Goals, and Understanding.* Hillsdale NJ: Lawrence Erlbaum Associates, 1977.

Thomas, F. and Johnson, O. *Disney Animation: The Illusion of Life* (Abbeville Press, New York, 1981).

Tu, X. and Terzopoulos, D. Artificial Fishes: Physics, Locomotion, Perception, and Behavior, *Computer Graphics* (Annual Conference Series, ACM, 1994) 43–50.

Webber, B., Badler, N., Di Eugenio, B., Geib, C., Levison, L. and Moore, M. Instructions, Intentions and Expectations, *Artificial Intelligence Journal* **73** (1995) 253–269.

Wilhelms, J. and Skinner, R. A 'Notion' for Interactive Behavioral Animation Control, *IEEE Computer Graphics and Applications* **10(3)** (1990) 14–22.

Wilkins, D.E. *Practical Planinng.* Palo Alto CA: Morgan Kaufmann, 1988.