Hierarchical Parallel Markov Models of Interaction

Robert Zubek and Ian D. Horswill

Northwestern University Computer Science
1890 Maple Ave, 3rd floor, Evanston, IL 60201
rob, ian@cs.northwestern.edu

Abstract

Finite state techniques are popular in entertainment software production, but they complicate the modeling of certain aspects of social engagement. In this paper we examine the problem of building probabilistic finite-state interaction models that allow both hierarchical composition of behaviors, and their parallel engagement. Finally, we propose an extension that resolves the difficulties for a class of common cases.

Interaction Modeling

Intelligent game agents must be able to interact with players, to present themselves as actual inhabitants of the virtual world. Popular approaches to interaction, such as static conversation trees and exchanges of pre-recorded blurbs, are widely recognized as insufficient, but a more generative natural language interaction will require a different approach to the underlying representation.

Interaction modeling in games is constrained by the demands of authoring, believability, and performance. The underlying mechanism must allow for easy behavior authoring, because developers require precise control over what the system will do at runtime, in order to provide enjoyable and engaging user experience, with specific aesthetic effects (Hunicke, LeBlanc and Zubek 2004). Additionally, developers want the behavior of the system to be believable, resembling the relevant aspects of human activity; ideally, believable behavior should be produced out of patterns matched against the game state, rather than being completely hand-scripted by a writer. Finally, the mechanisms must be computationally efficient, producing desirable behavior in only a fraction of total compute cycles.

For these reasons, simple techniques such as finite state machines (FSMs) remain very popular. Hand-crafted FSMs are routinely used to implement character behavior, since they are simple to author, intuitive to debug, and efficient to compute. In character interaction and conversation, simple dialog trees are very popular.

In domains where the inputs are noisy or ambiguous—such as in tracking human activity or language processing—stochastic generalizations are often used instead. Hidden Markov models (HMMs), for example, improve robustness by allowing for uncertainty; they continually re-estimate the probability of being in a given state, based on observed inputs.

The Hierarchical Parallel Problem

Rich, believable communication requires engagement in different social protocols, and broad competence in human conversational moves. Implementing the system using a large dialog network is clearly problematic. First, it is an authoring nightmare: finite-state models with non-trivial numbers of spaces are difficult to design and debug. Second, probabilistic implementations become computationally expensive, because HMM belief computation is superlinear in the number of states.

We can reduce the complexity of the system by decomposing the FSM into smaller, coupled FSMs. Consider a typical fantasy role-playing game. When building a shopkeeper agent, for example, we can decompose the selling interaction into the threads of deciding on an item, evaluating the item, haggling about the price, and so on; the evaluation thread can consist of praising the workmanship, pointing out features, or defusing criticism; praising the workmanship can then finally bottom out in particular speech acts. The decomposed system would be more efficient. And since individual modules can be developed independently and reused, it should also be cheaper to develop.

A natural implementation of such decomposition would be as an augmented transition network (ATN) or pushdown automaton (PDA), which are essentially FSMs that can "call" other FSMs that later "return" to the original FSM.

However, in real communication, multiple interactions can be interleaved concurrently. For example, consider the barter example above. Barter includes the child protocols of *haggling* and *praising*, simplified versions of which are shown as the linear chains **H** and **P** in Figure 1. In practice, these protocols are often, but not always, interleaved, as in: **P1 H1 H2 P2 P3 P4 H3 H4.** To follow the progress of the barter, the system must therefore be able to accept arbitrary interleavings of these sequences

without having to merge them into one (very large) FSM containing separate paths for every possible interleaving.

We therefore want our system to support not only hierarchical representations of social protocols, but parallel instantiations of them. Unfortunately, PDAs do not support concurrent execution. In the remainder of this paper, we will show how parallel and hierarchical structure can be modeled using a set of parallel, coupled, Markov models (essentially probabilistic FSMs). This allows flexible, robust interaction using a small number of computationally efficient, easily authored finite-state models.

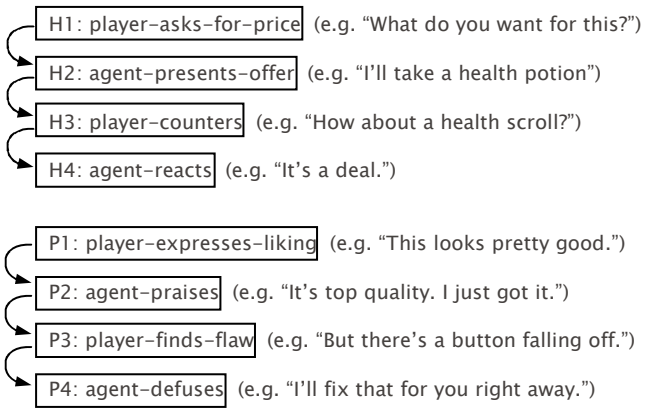

Figure 1. Sample threads.

HMM Essentials and Extensions

We implement coupling through extensions to the hidden Markov model (HMM) estimation. HMMs are effectively FSMs that keep track of a probability distribution over all possible states rather than a single "current state". While somewhat more expensive than FSMs, HMMs are much more robust when the input is insufficient to unambiguously determine the current state of the system.

The formal definition is as follows. Let $\mathbb{P} = [0, 1]$ be the subset of $\mathbb{R}$ representing valid probability values. A hidden Markov model $\mathbf{M}$ is a tuple $\langle S, A, p, q \rangle$, where:

- $S = \{ s_1, s_2, \ldots \}$ is the set of possible states, with a unique initial state s_1,
- $A = \{ \gamma_1, \gamma_2, \ldots \}$ is the set of discrete communicative actions (speech acts, etc.) that can be observed,
- $p : S \times S \to \mathbb{P}$ gives the probability of transitioning from one state to another, and
- $q : S \times A \to \mathbb{P}$ gives the probability of observing a given action when the system is in a given state.

To indicate the semantics of the probability functions, we will write $q(\gamma \mid s)$ as the probability of action γ being observed at state s, and $p(s' \mid s)$ as probability of transition to s' from s, where $\forall s' \in S: \sum_{s \in S} p(s' \mid s) = 1$.

Note that this defines action observation as dependent on state, rather than state transitions. These two representations are equally valid and easily convertible, using methods such as those described by Jelinek (1997). For our purpose, however, state-based observation will simplify the description.

Estimating the Current State

The precise "state" of interaction is not observable at any given time. However, an HMM allows us to estimate it based on a history of observations.

The belief *distribution* over all states is a function $b : S \times \mathbb{Z} \to \mathbb{P}$, written as $b_t(s)$, which is the system's estimate of the probability of the process being in state s at time t. The belief can be computed using popular forward algorithm for hidden Markov models (Jelinek 1997, Jurafsky and Martin 2000) extended to the state-based observation model:

$$b_t(s) = \sum_{s_i \in S} p(s \mid s_i)\, q(\gamma_t \mid s)\, b_{t-1}(s_i)\, \eta \qquad (1)$$

Here η is a normalization value use to force b_t to sum to 1. Furthermore, for the initial time slice t = 1, define $b_t(s_1) = 1$, and $b_t(s) = 0$ for all $s \neq s_1$. We will not discuss how to interpret the belief distribution function here, but rather jump right into the extensions—the interested reader is encouraged to reference Zubek (2004) or Roy et al. (2000) for more details.

When the system has to choose actions based on belief state, then the HMM is called a partially observable Markov decision problem or "POMDP". We then assume there is a policy $\pi : (S \to \mathbb{P}) \to A$ that maps belief states to actions.

Extensions

We will build the system from a collection of concurrent HMMs, coupled to model the hierarchical dependencies between the different parts of the model. This is done by extending the approach, to move elements unnecessary for state estimation out of belief computation. We now look at the technical details of the coupling mechanism, and the following section will detail how to use it to implement certain kinds of hierarchical dependence.

Action Observation Decomposition

The belief estimation equation (1) requires that the function $q(\gamma \mid s)$ be provided. Typically, its computation is treated as atomic. However, we can use Bayes' rule to

decompose it into two simpler functions and the normalization value η:

$$q(\gamma \mid s) = e_t(\gamma)\, g(s \mid \gamma)\, \eta \qquad (2)$$

The function *e* represents evidence estimation: it is the confidence level that some communicative action γ was actually observed from the system input at the given time, independently of the state. It categorizes the current input into likely speech acts, interpersonal actions, and other expression types, based on state-independent surface features. For example, the probability of a request should be high for a phrase beginning with "can you", but rather low if it begins with "you can". The function is really a probability distribution over all speech acts, and it can suggest multiple non-zero values: a phrase such as "yeah" could be simultaneously interpreted as a possible confirmation, as a positive response to a yes-or-no question, as a turn-taking acknowledgement, and so on. Estimators can be implemented using a simple, external pattern-matching mechanism.

The expectation function *g* then ties *e* to the specifics of the situation; it is the likelihood of observing that type of communication at the given state. For example, given the observation of a request, *g* specifies whether a request is likely in the given situation. This way, a protocol can specify which communicative acts accomplish movement through its different stages.

The multiplication of *e* and *g* accomplishes ambiguity resolution. Conceptually, *e* represents the likely interpretation of the input utterance, while *g* specifies if the interpretation makes sense for that state of conversation; their multiplication returns zero when either the input fails to fit the interpretation, or the interpretation is unexpected. The resulting value of *q* thus represents the *intersection of what was recognized with what was expected.*

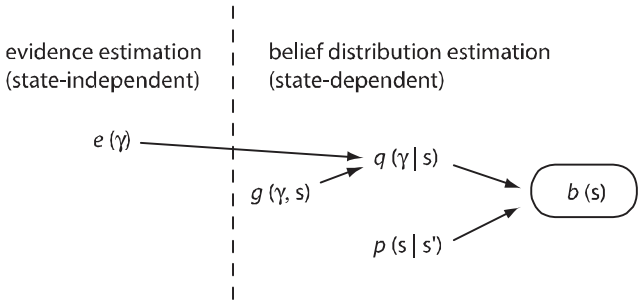

Figure 2. Functional view of belief distribution computation.

This separation may seem unnecessary, but its economy of expectation modeling aids the designer, decoupling input category recognition from its interpretation. The separation also achieves a useful architectural simplification. Evidence estimation is independent from states; therefore it can be computed separately from belief estimation, as seen in Figure 2. This will assist us in abstracting out dependencies.

Model Coupling

Using this separation, we now present an inspection mechanism for coupling the independent Markov models. Inspection allows the state belief distribution over one model to serve as evidence for belief calculation in another model in the system.

Define it formally as follows. Given two HMMs **A** and **B**, with states $s_A \in S_A$, $s_B \in S_B$, for each case when s_A *inspects* s_B, we augment A_A with a unique action γ' that signifies the inspection, and define $e_t(\gamma') = b_{t-1}(s_B)$, and $g(\gamma', s_A) = 1$. This makes the belief of the state s_A dependent on the belief in s_B from the previous time slice, but without actually joining the spaces.

Topic Tracking

In most interactions, there is additional state information, such as the object being bartered for, which is not easy to represent within the state graph of an FSM. For example, in the conversational robot by Roy, Pineau and Thrun (2000), the conversation includes separate states for the different TV stations one could talk about, such as *want-NBC-info*, *want-CBS-info*, or *want-ABC-info*; for the different locations where the robot could be ordered to go, such as *send-robot-to-kitchen*, *send-robot-to-bedroom*, and so on. This is necessary because FSMs are ultimately propositional representations, meaning they cannot use variables. A common resolution is to represent variable and value combinations directly as states in the state space. However, the number of possible combinations can get unwieldy.

In our work, we implement a different approach to simple variable binding, using external variables linked to evidence estimation. We track a small number of task-specific variables such as the current topic, the current item being bartered, etc., and represent them deterministically. Binding one of these variables then becomes one of the possible actions the system can perform. This is similar in spirit to Agre and Chapman's deictic representation (1987) in that there are only a few variables that are bound to values through actions. However, in deictic representation, those variables are implemented by perceptual attention.

The variable bindings are available for inspection by estimators, to determine the belief states of Markov models. For the details of their implementation, as well as

additional architectural considerations, please consult the forthcoming technical report (Zubek 2005).

Hierarchical Dependencies as Coupling

Finally, we describe how to model hierarchical control using model coupling. We would like to identify the following two forms of causal dependencies as salient for hierarchical models:

1. State dependence. This is when the state of one model influences the state of another. For example, a purchase interaction may need to be rolled back into a previous state, if haggling didn't succeed.
2. Model dependence. This is when the state of one model engages or disengages an entire other model. For example, starting an evaluation could enable a number of specialized behaviors, which get disabled once evaluation ends.

We reduce these dependencies to coupling, as follows.

State Dependence

State dependence happens when state belief in one model influences state belief in another model. Consider two states, the controlling state $s_C \in S_C$, and the target state $s_T \in S_T$, such that the belief in the controlling state is intended to influence the belief in the target state.

We eliminate this dependence via inspection. Set s_T to inspect s_C via some modification function f: define some unique action γ_C to correspond to the inspection, and define $e_t(\gamma_C) = b_{t-1} \circ f(s_C)$, and $g(\gamma_C, s_T) = 1$. The result is that the target's belief value will be a function of the controller's previous belief value.

Model Dependence

Model dependence happens when the state of one interaction can influence the engagement in an entire other interaction, as in push-down hierarchical models.

We eliminate this dependence via inspection. Consider the target state space S_T and some controlling state s_C belonging to a different model. Allow S_T to contain a unique disable state s_0, with incoming links from all states, and an outgoing link to the initial state s_1; belief distribution over s_0 signifies confidence that the entire model is disengaged.

Specify that s_0 inspects s_C in two ways: there exist two unique actions γ_E, γ_D that correspond to the engagement and disengagement of the submodel, mediated through some function f. Define $e_t(\gamma_E) = b_{t-1} \circ f(s_C)$, $e_t(\gamma_D) = 1 - b_{t-1} \circ f(s_C)$, and expectations $g(\gamma_D, s_0) = 1$, and $g(\gamma_E, s_1) = 1$.

We then use these two dependency reductions of state and model dependence, to translate causally dependent models into independent, weakly coupled ones.

Results

The resulting approach implements a hierarchy of parallel, coupled HMMs, each of which is very cheap to track.

Two systems use this approach to implement conversational interaction. The first, a conversational "sim" titled *The Breakup Conversation,* is a parody of the conversation at the end of a romantic relationship. The player starts out as someone involved in a failed relationship, who decides to end it via internet chat, and the role of the soon-to-be-ex is played by the computer. Figure 3 presents a fragment of the system's model hierarchy.

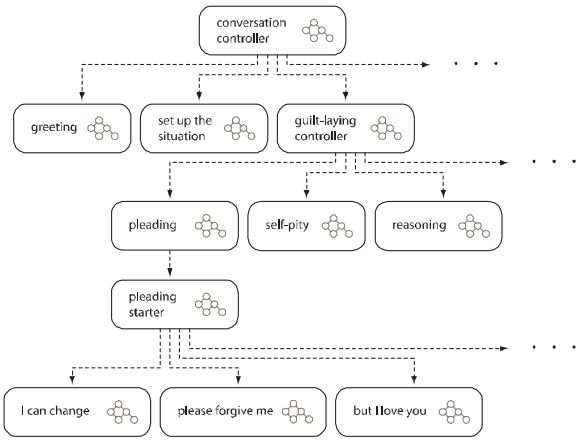

Figure 3. Fragment of the weakly coupled hierarchy from *The Breakup Conversation.*

The second system is *Xenos,* an NPC for the fantasy game *Neverwinter Nights,* which acts as a shopkeeper: sells items, introduces quests, or offers some bar goods, via natural language conversation. Its utterances are contextually coherent, but stylistically rough, to convey the idea that it does not really understand the language. An example of an exchange with Xenos is presented in Figure 4.

Xenos conversation fragment.
Computer's utterances and emotes in **boldface**:

Player says:	hi there
Xenos says:	**welcome**
Player says:	i need a new sword
Xenos glances	
Xenos says:	**i have for you sword**
Player says:	how is it
Xenos says:	**are you looking for a job?**
Xenos says:	**it's a dwarven sword**
Player says:	how much
Xenos nods:	**hm**
Xenos says:	**i'll trade for a health potion**
Xenos glances	
Player says:	i don't have one
Xenos says:	**too bad**
Player says:	about that job though
Player says:	what is it
Xenos says:	**i need to get a letter from the mayor**
Xenos says:	**could you go bring it from the mayor of our town?**

Figure 4. Xenos conversation fragment, including quest and item threads.

Both systems contain redundant models of interaction, on different levels of generality—Table **1** presents an overview of the different HMMs from the two demonstration systems. All of these models are active at the same time, all the time, and monitor the ongoing interaction; if the player steps outside of some specific model's area of competence, for example by using an unexpected speech act, the interaction will fall back onto more general components, which will deal with the situation. The result is an interaction system that exhibits the desirable hierarchical parallel property: the interaction retains hierarchical composition of sub-interactions that can affect over each other, but at the same time allows for parallel and independent engagement on different levels of generality. The redundant representation softens the performance drop-off at the edge of competence, and can be used to mimic some of the human ways of dealing with error.

The systems are exceedingly efficient. A recent version of the *Conversation* is built on a set of 75 coupled HMMs and, as described in a forthcoming report (Zubek 2005), it takes less than 30ms on the average to process an utterance —including parsing, evidence estimation, belief computation, and output production. Indeed, most of the

Shared Spaces

Low level monitors:
- Conversation timer
- Silence monitor
- Monologue monitor

Ambient movement:
- Fidget machine
- Turn enforcement
- Ambient emote
- Stock response
- Turn monitor
- Topic monitor

Insult management:
- Direct insult
- Indirect insult
- Insult accumulation

General routines:
- Topic recognized but not the form
- Question recognized but not topic
- Question about object
- Question about health
- Request general
- Request item
- Disagreement
- Player evaluation
- Agent evaluation
- Condemnation monitor

Conversation structure:
- Greeting
- Intro conversation
- Outro conversation

Breakup-specific

Breakup intro:
- Allude to breakup
- Giving in monitor
- Guilting coordinator

Guilting: Self-pity
- Self-criticize
- Reject compliment
- "You must hate me"
- "Why are you mean"
- "Will you help me"

Guilting: Indignation
- "I thought you loved me"
- "I thought you cared"
- "I don't deserve this"
- "How can you do this"

Guilting: Pleading
- Beg for second chance
- Promise change
- "But I love you"

Guilting: Reasoning
- Guess at reason
- Demand reason
- Evaluate reason
- Treat as excuse
- Deny reason

Panicking:
- Silence
- Impatience monitor
- Resignation monitor
- Rejection monitor
- Start panic

Xenos-specific

Quests:
- Quest monitor
- Perform quest injection
- Deal with agreement
- Deal with rejection
- Rush the player

Special routines:
- Job request
- What question
- Where question
- Payment question
- Evaluate object
- Barter for object

Table 1. Outline of the different spaces used in the two implementations.

computation is spent running an off-the-shelf parser—without parsing, the HMM computation takes less than 0.5ms on the average.

In our experience, the separation of interaction elements into distinct models simplifies authoring, and the weak coupling limits their possible interactions, simplifying debugging. Finally, good preprocessing and compile-time optimizations make the HMM calculations very cheap. Act estimation can be implemented using inexpensive pattern matching, belief distribution can be optimized as matrix multiplication, and action production can be done as cheap template filling.

Related Work

Finite-state techniques are popular in dialog and interaction modeling, and "[m]ost commercially available dialogue systems use some form of finite-state dialogue modeling" (McTear 1998). Cole et al. (1995) present a good overview of these approaches, in the *Survey of the State of the Art in Human Language Technology,* with more detailed background information is available in Grosz et al. (1986).

Stochastic techniques have been popular in language modeling on the utterance level (Jelinek 1997, Roche and Schabes 1997), and have been recently extended to dialog modeling (Pineau, Roy and Thrun 2001, Young 1999, Levin and Pieraccini 1997). Hierarchical HMMs (Fine, Singer and Tishby 1998) and related approaches (Ghahramani and Jordan 1995, Saul and Jordan 1995) are currently being investigated for dialog engagement.

Related work from entertainment includes developments on managing dialog as part of a larger action selection architecture (Loyall 1997, Mateas and Stern 2002). Also, entertainment products routinely employ pattern matching techniques such as those used in *chatterbots* (Mauldin 1994, Hutchens 1998).

Acknowledgements

Special thanks to Robin Hunicke, Praveen Paritosh, and Ayman Shamma for their comments on this work.

References

Cole, R. A., Mariani, J., Uszkoreit, H., Zaenen, A., Zue, V., eds. 1995. *Survey of the State of the Art in Human Language Technology.* National Science Foundation, et al. Reprinted at Cambridge: Cambridge University Press, 1986.

Fine, S., Singer, Y., Tishby, N. 1998. The Hierarchical Hidden Markov Model: Analysis and Applications. *Machine Learning,* 32: 41.

Ghahramani, Z., and Jordan, M. 1995. Factorial Hidden Markov Models. *Proceedings of the Conference on Advances in Neural Information Processing Systems (NIPS),* vol. 8, pp. 472-478.

Grosz, B. J., Jones, K. S., and Webber, B. L., eds. 1986. *Readings in Natural Language Processing.* Los Altos, CA: Morgan Kaufmann.

Hunicke, R., LeBlanc, M., Zubek, R. MDA: A Formal Approach to Game Design and Game Research. *Proceedings of the AAAI Workshop on Challenges in Game AI,* AAAI Tech Report WW-04-04. Menlo Park, CA: AAAI Press.

Hutchens, J. 1998. Introducing MegaHAL. *NeMLaP / CoNLL Workshop on Human-Computer Conversation.* Association for Computational Linguistics.

Jelinek, F. 1997. *Statistical Methods for Speech Recognition.* Cambridge, MA: MIT Press.

Jurafsky, D., Martin, J. H. 2000. *Speech and Language Processing.* Englewood Cliffs, NJ: Prentice Hall.

Levin, E., Pieraccini, R. 1997. A Stochastic Model of Computer-Human Interaction for Learning Dialogue Strategies. *Proceedings of Eurospeech '97,* pp. 1883-1886. Rhodes, Greece.

Loyall, A. B. 1997. *Believable Agents: Building Interactive Personalities.* PhD Dissertation, School of Computer Science. Pittsburgh: Carnegie Mellon University.

Mateas, M., Stern, A. 2002. *Architecture, Authorial Idioms and Early Observations of the Interactive Drama Façade.* Technical Report CMU-CS-02-198, School of Computer Science. Pittsburgh: Carnegie Mellon University.

Mauldin, M. 1994. Chatterbots, TinyMUDs, and the Turing Test: Entering the Loebner Prize Competition. *Proceedings of AAAI-94.* Menlo Park, CA: AAAI Press.

McTear, M. F. 1998. Modelling spoken dialogues with state transition diagrams: experiences of the CSLU toolkit. *Proceedings of the International Conference on Spoken Language Processing,* vol. 4, pp. 1223-1226. Sydney: Australian Speech Science and Technology Association, Incorporated

Pineau, J., Roy, N., Thrun, S. 2001. A Hierarchical Approach to POMDP Planning and Execution. *Workshop on Hierarchy and Memory in Reinforcement Learning (ICML).* Williams College, MA.

Roche, E., Schabes, Y. 1997. *Finite State Language Processing.* Cambridge, MA: MIT Press.

Roy, N., Pineau, J., Thrun, S. 2000. Spoken Dialogue Management Using Probabilistic Reasoning. *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics.*

Saul, L. K., Jordan, M. I. 1995. Boltzmann Chains and Hidden Markov Models. *Proceedings of the Conference on the Advances in Neural Information Processing Systems (NIPS),* vol. 7. Cambridge, MA: MIT Press.

Young, S. 1999. Probabilistic Methods is Spoken Dialogue Systems. *Proceedings of the Royal Society,* September 1999. London.

Zubek, R. 2004. Character Participation in Social Interaction. *Proceedings of the AAAI Workshop on Challenges in Game AI,* AAAI Tech Report WW-04-04. Menlo Park, CA: AAAI Press.

Zubek, R. 2005. Forthcoming Ph.D. dissertation, Computer Science Department, Northwestern University.