Towards Situated Activity Management: Representation, Inference and Decision Making

Dan Bohus, Ece Kamar and **Eric Horvitz**
One Microsoft Way
Redmond, WA, 98052
{dbohus,eckamar,horvitz}@microsoft.com

Abstract

We introduce and discuss the problem of *situated activity management*, and present an approach for guiding and coordinating an agent's speech and activities amidst the dynamics of evolving settings. Specifically, we couple a hierarchical representation of activities with a novel state tracking approach based on conditional Markov Networks. We show how the approach can enable an agent to reason jointly about parallel coordinated actions and the changing situational context. The hierarchical structure and joint inference allows for a modular authoring of rules used for making in-stream decisions. We have implemented a proof-of-concept for this approach and illustrate its functionality with a simulated dialog trace.

1 Introduction

Recent research on developing physically situated interactive systems highlight spoken dialog as only one component of the larger challenge of managing parallel, coordinated actions amidst a dynamically changing world. As an example, consider a robot in charge of greeting, interacting with, and escorting visitors in a building. To plan and generate appropriate utterances and actions, such physically situated agents need to reason continuously about the *situational context*, *e.g.* people, objects, events in the environment, as well as their relationships and temporal dynamics. The context often evolves asynchronously with respect to the utterances spoken and the actions executed, and therefore interaction planning must be done *continuously, in stream*, rather than on a per-turn basis. Managing interaction in these settings requires reasoning about complex patterns of *coordination between parallel actions* (linguistic and non-linguistic) produced by multiple participants.

We introduce and discuss the problem of *situated activity management*, and argue for broadening the scope of work in dialog management along the lines described above. Second, we introduce a candidate representation and associated inferential methodology for situated activity management. Specifically, we couple a hierarchical representation of activities and actions with a novel approach to situated state tracking based on conditional Markov Networks and a tractable mechanism for action selection. The approach allows a modular authoring process and enables an agent to reason about the situated state, and to make in-stream decisions about actions. The methodology builds upon and extends several strands of research on hierarchical representations for multi-agent planning and belief tracking under uncertainty.

While an extensive comparison with existing dialog management techniques (*e.g.*, Allen et. al, 2001; Churcher et al., 1997; McTear, 2002, Traum and Larsson, 2003) falls beyond the scope of this paper, we begin by discussing relationships with the most relevant prior work, and framing the novelty of the proposed approach. We then formally introduce the broader problem of situated activity management in Section 3 and present the methodology in detail in Section 4. In Section 5, we illustrate the proposed approach with a simulated trace, and in Section 6 we conclude and present directions for future work.

2 Related Work

Hierarchical representations are a natural choice for modeling the structure of human activities and interactions. Previous work (Grosz, 1974; Grosz and Sidner, 1986) demonstrated that task-oriented

dialog has a nested structure that often mirrors the hierarchical structure of the task at hand. Subsequently, several hierarchical formalizations have been proposed for modeling collaboration and discourse (Grosz and Kraus, 1996; Lochbaum, 1994) and have been brought into practical use in dialog management, *e.g.*, in Collagen (Rich and Sidner, 1997) and RavenClaw (Bohus and Rudnicky, 2009). Hierarchical representations have useful properties, such as modularity, scalability, transparency, and ease of authoring. They also provide a natural basis for handling discourse phenomena like shifts of focus and nested sub-dialogs. However, methods based on hierarchical representations have not generally addressed the uncertainty that characterizes real-world systems equipped with noisy sensors and recognizers.

The problem of managing uncertainty is of great practical importance, and as such has received a great deal of attention in the spoken dialog community. Significant research efforts have been targeted at modeling dialog as a partially observable Markov decision process (POMDP) (*i.a.*, Williams and Young, 2007), and at learning optimal policies from data. While advances have been made in increasing the scalability of these approaches (Williams and Young, 2005; Williams, 2010), real-world applica-tion of reinforcement learning and POMDP techniques has been limited. The approaches do not inherently make use of the hierarchical nature of dialog and, even with factored representations, state inference and action selection can become in-tractable. Practical dialog applications using POMDPs have been limited because of challenges with transparency, maintainnability, and definitions of reward function and optimization criteria (Paek and Pieraccini, 2008).

We seek to bridge the gap between hierarchical representations and probabilistic methods for dialog by integrating belief tracking and decision making under uncertainty into a hierarchical formalization of situated activities. Our goal is to retain the modularity, ease-of-authoring, scalability, and maintainability properties that hierarchical representations confer, and combine them with an efficient mechanism for reasoning and decision making. Moreover, the proposed methodology targets a challenge broader than traditional dialog management, and enables reasoning about parallel coordinated actions and making decisions in-stream, rather than on a per-turn basis.

3 Situated Activity Management

We begin by defining the problem of *situated activity management* and discussing a set of core competencies required for managing situated activities under uncertainty.

By *activity* we denote a *set* of atomic *actions* that are executed in the world by the participants of the activity. Actions have a start and an end time, and a set of property values. In the case of spoken dialog, utterances are modeled as *actions* and discourse segments are modeled as *activities*. For instance, in the robot example, the utterance *"My name is Paul"* can be modeled as an action: IAm(Name=Paul). The sub-dialog for getting directions can be modeled as an activity. Actions can extend however beyond the linguistic domain, and may correspond to any atomic action performed in the world, *e.g.*, opening a door, pointing, or asking a question. The actions that form an activity are generally coordinated: their content and timing depends on the actions previously executed by all participants. Multiple actions may be executed in parallel, and may overlap each other. In Figure 1, the joint set of actions being executed at time t are represented by A_t.

By *situated* we refer to the fact that the activity occurs in a broader *situational context*, which also influences the execution of actions. The situational context is the state of the world that is relevant to the ongoing activity. The situational context may evolve asynchronously with respect to the actions being performed. Its dynamics may depend on the actions executed, but also on other external factors that are outside the control of the participants involved in the activity. In the robot domain, the context may include static variables such as the identity of the user and dynamic variables such as the current location of the robot, etc. In Figure 1, the situational context at time t is represented by C_t.

A *situated activity management model* enables an agent to engage in and contribute to a situated activity with other agents. We seek to construct such a model and take the decision-theoretic perspective of an agent acting under uncertainty: we assume the agent does not have access to the ground truth about the actions executed or about the context. Instead, the agent has to reason from noisy, streaming observations about the context and actions being produced (denoted by O_t in Figure 1). For instance, in our example, the robot might have access to location sensors, face recognition sensors, etc.

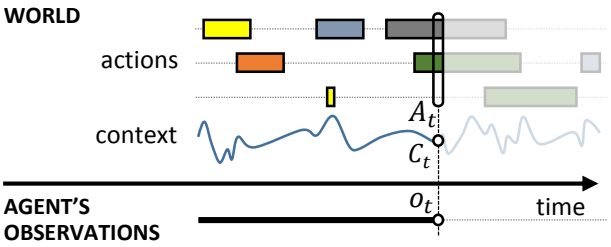

Figure 1. Ground truth and observations for actions and context during a situated collaborative activity

Two core competencies are required for managing situated activities under uncertainty. First, the agent must be able to track the situated state, based on incomplete and potentially noisy observations (o_t). This includes recognizing the actions and activities being executed (A_t), and tracking the situational context (C_t). Second, the agent must be able to reason under uncertainty about which actions, if any, to execute, so as to maximize the utility of the activity at hand.

4 Approach

The methodology we propose is centered on a hierarchical representation which encodes how activities can be decomposed hierarchically into sub-activities and actions, and captures the variables relevant for making decisions in each activity. We couple this representation with an inference mechanism that jointly tracks the situated state based on an undirected graphical model. The representation and inference mechanism enable a tractable, modular approach to decision making. In the following subsections, we discuss the representation, inference, and decision making components in more detail.

We begin by introducing a simplified example problem, inspired by an embodied conversational agent that serves as a personal assistant outside the office door of an employee at our organization (Bohus and Horvitz, 2009). The Assistant, has access to streaming information and predictions about the owner's location, calendar, and proximity, and uses this data along with evidence gleaned from multimodal scene analysis to schedule meetings on behalf of the owner with people that stop by when the owner is not present. To do so, the Assistant first has to identify the user, by relying on a face recognition algorithm, or by asking the user who they are. Alternatively, it might ask the user to swipe their badge on an RFID badge scanner. Once the user has been identified, depending on the owner's availability, the Assistant either suggests a meeting right away or tries to schedule one for a later time appropriate for the user and the owner.

4.1 Representation

The building block of the hierarchical representation is the *activity model*. Activity models are authored by system developers and define how an activity decomposes into sub-activities and actions. At runtime, they are automatically combined into an *activity tree*, which acts as a backbone for inference and decision making. Non-terminal nodes in the activity tree represent activities, while leaves represent actions. Figure 2 illustrates the activity tree for the Assistant, formed by combining six activity models. At the top-level, AssistGuest decomposes into the IdentifyGuest, ScheduleMeeting and MeetNow activities. Below, IdentifyGuest decomposes into GetName and GetId, and so on.

Each activity model defines a set of *state variables* and *observations* that are relevant for making decisions about that activity. State variables model the information that is not directly observable to the agent, and must be inferred based on available observations. For each activity we introduce a state variable to capture whether and how that activity is being executed at the current moment. For instance, the *GetName*, *WhoAreYou*, *IAm* variables capture the execution of the corresponding activities in the GetName model – see also Figure 3. For actions or activities that have parameters, e.g. IAm(name=x), the corresponding variable, e.g. *IAm*, spans the set of possible action instantiations, e.g. IAm(name=Eve), IAm(name=Paul), etc., plus a value ∅ indicating that the activity is not currently being executed.

Each activity model may also define variables that capture contextual information that is relevant for performing the activity. For instance, the *name* variable in the GetName model captures the user's name, which is relevant to how the user will perform the IAm action: a user named Paul is likely to perform IAm(name=Paul) rather than IAm(name=Eve). While the name is static, other context variables might be dynamic and evolve asynchronously with respect to the interaction. For example, a *proximity* variable in the AssistGuest activity captures whether the owner is close by or far away, and is relevant for the agent's choice between suggesting an immediate meeting or scheduling one for later.

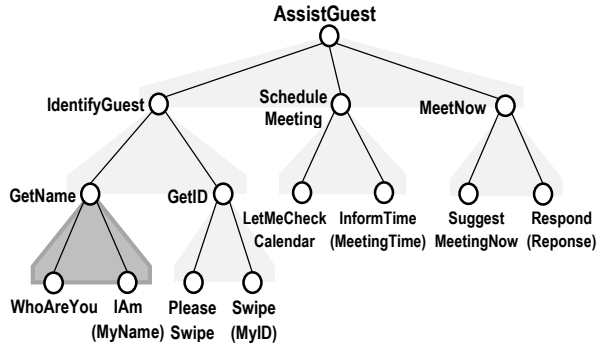

Figure 2. Multiple activity models comprise an activity tree

Finally, *observations* defined in each activity model capture the evidence about action execution and the situational context, and they are used to infer the current state. In our example, a *faceid* observation in the GetName model arrives from a visual face recognition component and can be used to make inferences about the user's name. In the same model, o_{IA} observation is obtained from the speech recognizer and can be used to make inferences about the IAm action.

The activity models specify the variables that are relevant for each activity and therefore enable modular authoring. Independently authored models are combined together through the variables that are shared by neighboring models to facilitate reasoning over the entire activity tree. When non-neighboring models share a variable, the variable is replicated in the models connecting them with identity relationships to enable joint inference (see *name* variable in Figure 3).

4.2 Inference

The goal of the inference mechanism is to jointly track the action execution and context variables through time, based on the observations collected so far (see Figure 1). We accomplish this with a conditional Markov Network (Koller and Friedman, 2008) induced over the activity tree. Markov Networks have become a popular choice for solving real-world joint inference problems in areas such as vision, bioinformatics, natural language processing.

A Markov network (also referred to as a Markov field, undirected graphical model, or factor graph), is a model for a joint probability distribution over a set of random variables $X = \{X_i\}$. The graph for a Markov network consists of a set of *nodes*, one for each variable of interest X_i and a set of *potentials* $\phi_k(V_k)$ that capture probabilistic relationships among subsets of variables $V_k \subset X$. Specifically, a potential ϕ_k over a set of variables $V_k = \{X_1..X_p\} \subset X$ is a function $\phi_k: \mathcal{D}(V_k) \rightarrow R^+$ that, for any joint assignment of the variables in V_k outputs a positive real number capturing the relative strength of that assignment. The joint distribution of the variables in a Markov network is expressed as the normalized product of potentials:

$$P(X) = \frac{1}{Z}\prod_k \phi_k(V_k)$$

where Z is a normalization constant, also referred to as the partition function. When a subset O of the variables in X are observed, we have a *conditional* Markov network, which models the conditional distribution $P(X|O)$. In this case, the partition function is conditioned on the observations.

As we have already seen, the activity models define the set of state variables and observations relevant for performing an activity. In addition, they also define the set of *potentials* that capture the relationships among these variables. Potentials may be represented in parametric form or in full tabular format, may be manually authored or learned from data, and can capture various types of stochastic or deterministic relationships between variables and observations. Two types of potentials may be defined by each activity model: *time-slice potentials* and *transition potentials*.

Time-slice potentials capture relationships between variables and observations within a given time step. Figure 3 illustrates some of the time-slice potentials from our example. For instance, the potential $\phi_1(name)$ is defined exclusively over the *name* context variable and captures a prior over it. The $\phi_2(name, faceid)$ potential links the *name* variable with a corresponding *faceid* observation. $\phi_3(name, IAm)$ models the relationship between *name* to the *IAm* action execution variable: it indicates that the instantiation of the *IAm* action will correspond to the name, or will be $\emptyset$, *i.e.*, not-executed. $\phi_4(GetName, WhoAreYou, IAm)$ is a three-way deterministic potential that links the action execution variables in the GetName model and indicates that GetName is being executed when either WhoAreYou or IAm is being executed.

*Transition potential*s model the temporal dependencies between variables and enable tracking of the situated state over time. For instance, in the example, a transition potential is defined over the

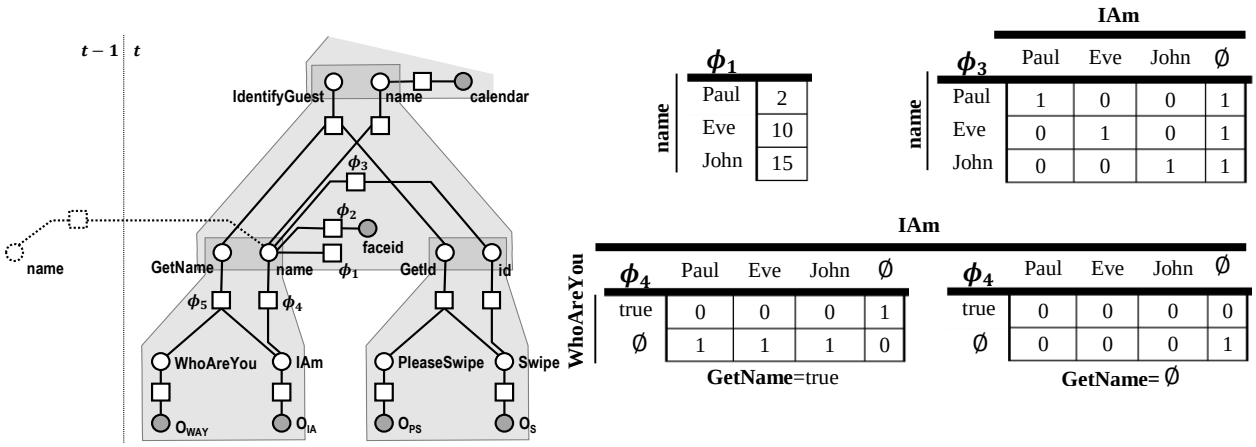

Figure 3. Left: Markov network corresponding to a portion of the AssistGuest activity tree. Right: Three example potentials.

name variable. As the name is static, this potential encodes an identity relationship.

At runtime, the variables, observations and potentials of different models are assembled to form a dynamic conditional Markov Network, which is used to track the situated state over time (see Figure 3). At each new time step t, the network is extended by coupling a new instantiation of the network induced by the activity tree for the next time step. The inference performed over the network is *joint*: it combines observations collected about actions and context throughout time, and from different parts of the activity tree, to compute the most informed beliefs about the current situated state. For instance, as we shall illustrate in Section 5, an observation of the Swipe action in the GetId model can combine with the noisy observation of a parallel IAm action in the GetName model, and with the face recognition observation of the IdentifyGuest model to infer the value of the *name* variable, and resolve which instance of the IAm action is being performed.

Inference algorithm

Inference in dynamic conditional Markov Networks is a challenging task: in the worst case, exact inference is exponential in the number of variables in an activity tree (Koller and Friedman, 2008). We can exploit however the hierarchical, structured nature of the proposed representation to provide a customized, more scalable approach.

For simplicity of presentation, we begin by considering inference within a single time slice and later extend to temporal inference. Except for very small size networks, exact inference is not feasible in general Markov networks that may have large cliques (loops or potentials connecting many variables). Approximate inference techniques like loopy belief propagation or sampling approaches would be required, which may introduce unbounded errors. The conditional Markov Network induced by the composition of activity models at a single time-step has however a special, hierarchical, tree-like structure, in which cliques are contained within individual activity models and the size of each clique is limited by the number of variables in the corresponding activity model rather than the size of the entire tree. This structure can be leveraged to speed up inference. Here, we take an approach where we collapse the variables within an activity decomposition model into a single joint variable. With the new structure, the network becomes a tree, and exact inference can be performed using belief propagation in time exponential in the number of variables in the largest activity model, rather than in the total number of variables in the activity tree. This novel approach results in exponential savings in the complexity of inference over an activity tree composed of a large number of activity models.

Temporal inference raises additional significant challenges as the inference at time t must be conditioned on the full joint of the variables at the previous time step, which is again exponential in the number of variables in the activity tree. In most cases, only a subset of variables, $S \subset X$, has transition potentials. In addition, collected observations may render only a small subset of joint configurations of transition variables likely. Based on these observations, we couple our within time-slice belief propagation approach with a particle-based approach to address the intractability of

temporal inference. Specifically, we represent the joint over the situated state at time $t-1$, with a set of n_{t-1} particles $\{s_i^{t-1}\}$, and their corresponding weights $\{w_i^{t-1}\}$. Each particle s_i^{t-1} represents a particular assignment of the state transition variables at $t-1$, and w_{t-1}^i captures the probability of this assignment, $w_i^{t-1} = P(s_i^{t-1}|o_{1:t-1})$.

To infer the state at the time step t, we project each particle s_i^{t-1} forward through the transition potentials and run a separate time-slice belief propagation inference on the activity tree for each incoming particle. In each of these inferences, the previous time-step variables on the transition potentials become observations, and are assigned to the values dictated by the incoming particle s_i^{t-1}. The belief-propagation based approach for time slice inference is then applied on each instantiated graph. The approach computes the marginal of the each situated state variable X_j at time t, conditioned on the incoming particle and current observations $P(X_j^t|s_i^{t-1}, o^t)$. In addition the belief propagation algorithm on the tree structure allows us to efficiently compute the partition function corresponding to each particle – $Z_i^t(s_i^{t-1}, o^t)$. The overall marginal probability of each state variable X_j at time t, conditioned on all observations up to time t can then be computed by combining the inference results for each individual particle with the weight of that particle:

$$P(X_j^t|o^{1:t}) = \frac{\sum_i P(X_j^t|s_i^{t-1}, o^t) \cdot w_i^{t-1} \cdot Z_i^t(s_i^{t-1}, o^t)}{\sum_i w_i^{t-1} \cdot Z_i^t(s_i^{t-1}, o^t)}$$

Finally, transition particles for the next time step, representing the joint probability distribution at time t are generated by sampling values for each variable in the network. If the total number of particles capturing the joint distribution is small, this sampling can be exhaustive and the entire distribution will be covered. Alternatively, sampling can be guided to select high-likelihood particles according to the computed marginal and can be run until a large percentage of the total mass has been covered. In the former case, the inference is exact, while in the latter it is approximate.

The approach we have sketched above performs filtering in the dynamic Markov network: it computes the marginal of state variables at time t based on accumulated observations, *i.e.* $P(X_j^t|o^{1:t})$. When most transition potentials in a network encode deterministic relationships between variables, full inference over the interaction history, *i.e.* $P(X^{1:t}|o^{1:t})$ may become feasible. Particles can be extended to represent each possible interaction history. The representation also offers an important tool for writing decision rules easily by querying the entire interaction histories.

4.3 Decision making

The decision-making mechanism in the proposed model also leverages the hierarchical nature of the activity tree. The set of activities and actions to be performed by the agent are computed hierarchically, in a top-down manner. Each activity model defines an *action selection function* that computes the intentions to execute each of the sub-activities or actions in the activity model. These functions are designed by the system author and the decisions are conditioned on the inferred state. Decision rules are often conditioned on the marginal of a single variable of the activity. When a rule needs to be conditioned on the joint of all variables in an activity, this joint is already provided by our inference approach that collapses the variables in each activity model. Finally, when action histories can be efficiently represented with transition particles as discussed above, action selection can also be conditioned on the distribution over the possible discourse histories for ease of authoring.

The hierarchical decomposition of activities allows for authoring decisions in a modular fashion, and enables hybrid action selection approaches, where different mechanisms are used in different activities in the tree. For instance, in certain cases simple rules may suffice, *e.g.* "keep executing IdentifyName until the probability of name exceeds a threshold, then move on to ScheduleMeeting if the owner is far away or MeetNow now if the owner is close by". Another functionality supported by the formalization for action selection is allowing authors to design a function which quantifies utilities for executing a subset of sub-activities or actions, conditioned on a complete assignment of variables of an activity. To perform action selection, the system computes the expected utility of executing any subset of sub-activities and/or actions under the inferred distribution of variables, and chooses the subset that maximizes this expectation.

The proposed model thus combines knowledge that can most often be easily expressed by system

authors (decision rules given beliefs over the situated state and relationships between key variables) with computational machinery for joint inference within the hierarchical representation of activity trees.

5 An Example Trace

As a proof of concept, we have constructed an initial MATLAB implementation of the methodology including the inference approach described above. We have authored activity models and experimented with several interaction scenarios. The goal at this stage is to validate the proposed approach, investigate its feasibility, and study and demonstrate some of the functionalities it enables. We generated traces by providing simulated observations and logging the results of the inference and decision making mechanisms at each time step.

As an example, in Figure 4, we show a trace based on the previously introduced assistant domain. In this scenario, the agent engages with a visitor, by activating the intention to AssistGuest at time 3. The agent then computes its intentions hierarchically, in a top-down fashion using the action selection function in each activity. Here, since the top hypothesis for name is not known with high enough confidence (see Figure 4 at time 3), the implemented functions drive the AssistGuest model to select IdentifyGuest, which in turn selects GetName, which in turn selects WhoAreYou.

Between times 3 and 8 the agent maintains this intention to execute WhoAreYou. As the action is being produced in the world, the corresponding observation o_{WAY} allows the system to infer that this action is in progress. In the example, the agent can perfectly execute its intentions. We note however that, with the proposed model, the agent computes its intentions, and then observes the actual execution of its actions. Therefore, the formalism can also support reasoning about cases where the system execution does not perfectly correspond to its intentions (*e.g.*, because of failures in action execution), and enable the agent to act accordingly in such cases.

Throughout this time, the model continuously infers the visitor's *name* by fusing evidence from multiple sources: the prior ϕ_1 which captures historical statistics, the calendar observations ϕ_6 (guests with scheduled meetings are more likely to show up), and streaming *faceid* observations $\phi 2$.

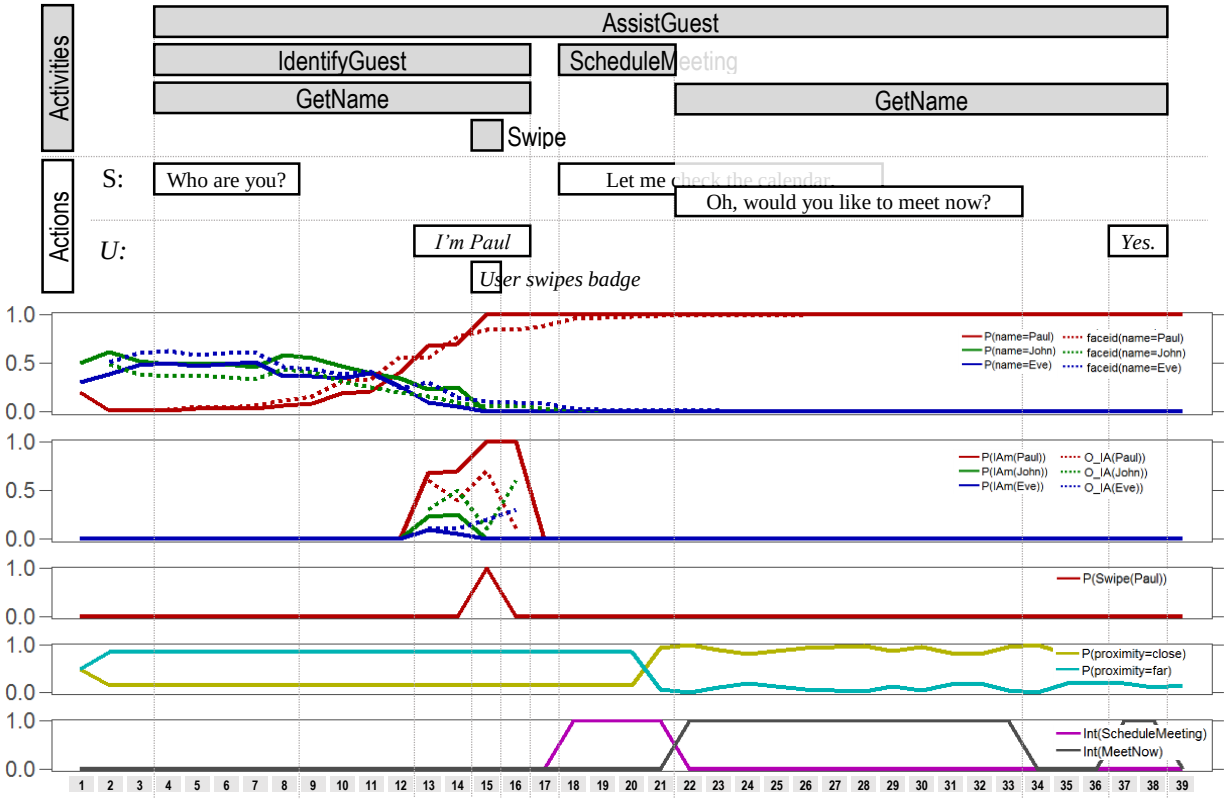

Figure 4. A simulated interaction trace

The *faceid* observations, and the inferred marginal over *name* throughout time are shown in Figure 4. Between times 13 and 16 the visitor says *"I'm Paul"* and the system receives corresponding partial hypotheses from the speech recognizer. At times 13 and 14, the partial recognition hypotheses are used in conjunction with the previous knowledge sources to update in-stream the inference on *name*. In parallel with speaking, the visitor swipes his badge at time 15. At this time, the observation from the badge swipe also contributes to inference and perfectly determines the *id* variable. As a result of joint inference, this observation also identifies the *name*, and the instance of the *IAm* action being performed. At shown in see Figure 4 at time 16, while the *faceid* observations and the speech recognition observation o_{IA} for the IAm action are still unconfident, but because of the badge swipe, the marginal over *name* and *IAm* action execution identifies that the person is Paul and the action performed is IAm(name=Paul).

At time 17, the visitor is not speaking and the system is confident about his identity. Consequently, the decision rule of the AssistGuest model identifies the ScheduleMeeting activity to perform next since the inference over the proximity of the owner indicates that the owner is not close by at the moment. The system launches the LetMetCheck-Calendar action at time 17. Moments later, at time 21, the inference over *proximity* is updated with new observations, and the decision rule in the AssistGuest model initiates a focus shift to MeetNow, suggesting that the guest meets with the owner immediately as the owner is about to arrive. When computing the intentions to execute within the MeetNow activity, the agent has access to the particle representation of the discourse history. As such, since it knows the ScheduleMeeting activity had already started the agent prepares the visitor for the shift to MeetNow by customizing the action and saying *"**Oh**, would you like to meet now **instead**?"* The visitor accepts meeting now and the interaction finishes.

While small, the example we have described illustrates several of the benefits and promise of the proposed model, such as joint inference and reasoning about parallel actions, as well as decisions driven by streaming observations. We are continuing to study other dialog and interaction phenomena in this formalism, such as focus shifts, disambiguation, action failures and repeated actions, etc.

6 Conclusion and Future Work

Motivated by some of the key challenges faced by physically situated interactive systems, we have articulated the problem of situated activity management. We have introduced a hierarchical representation of situated activities and an associated inference and decision making model. The proposed model aims to bridge the gap between hierarchical formalizations of collaborative activities and reasoning under uncertainty. It enables reasoning and making in-stream decisions about parallel, coordinated actions, and introduces the use of conditional Markov Networks as a novel tool for state tracking in dialog systems.

The model we have introduced is a first step in a larger endeavor. We believe that the approach lays a foundation for exploring in-stream interaction management in physically situated settings. Several research questions arise on the path towards the application of this methodology to large, real-world dialog settings. As stated earlier, joint temporal inference of the situated state is a challenging task. In addition to gains in efficiency afforded by the hierarchical representation, further research is needed on exploiting additional structure for tractability in problems that have variables with large domains and that do not compartmentalize variables to individual activity models (*i.e.*, share a lot of variables across many activity models). We are investigating compression-based approaches to efficiently represent large variable domains and approximate approaches for inference and for representing transition particles succinctly when they become very numerous.

In addition, we believe there are opportunities for automation in the authoring of potentials and decision rules that layer-in domain-independent models of engagement, turn-taking, focus-of-attention, and multiparty collaboration, over the manually authored domain-specific constraints and logic. Finally, we are experimenting with different dialog examples to investigate the generality of our representation and the limits of the inference and decision-making algorithms that we have outlined. We believe continued research efforts on the situated activity management problem and representation and methodology introduced here will lead to systems with rich interaction capabilities in physically situated settings.

Acknowledgments

We would like to thank Barbara Grosz and Ashish Kapoor for their suggestions and feedback in the development of this work.

References

Allen, J.F., Byron, D.K., Dzikovska, M., Ferguson, G., Galescu, L., and Stent, A. 2001. Towards Conversational Human-Computer Interaction, *AI Magazine*, **22**(3)

Bohus, D., and Rudnicky, A. 2009. The Ravenclaw dialog management framework: Architecture and systems, in *Computer, Speech and Language*, **23**(3).

Bohus, D., and Horvitz, E. 2009. Dialog in the Open-World: Platform and Applications, in *Proc of. ICMI'09*, Boston, MA

Churcher, G. E., Atwell, E.S, and Souter, C. 1997 Dialogue Management Systems: a Survey and Overview, *Technical Report*, University of Leeds, Leeds, UK.

Grosz, B., 1974. The Structure of Task Oriented Dialogs, in *Proc of IEEE Speech Symposium*, Carnegie-Mellon University, Pittsburgh, PA, 1974.

Grosz, B.J., and Sidner, C.L. 1986. Attention, Intention and the Structure of Discourse, *Computational Linguistics*, **12**(3), 1986

Grosz, B.J., and Kraus, S., 1996. Collaborative Plans for Complex Group Action. *Artificial Intelligence* **86**(2), 269-357.

Koller D., and Friedman, N. 2010. Probabilistic Graphical Models: Principles and Techniques, ISBN 978-0-262-01319-2

Lochbaum, K.E., 1994. Using Collaborative Plans to Model the Intentional Structure of Discourse, *Technical Report TR-25-94*, Harvard University, Ctr. for Res. in Computing Tech. PhD Thesis.

McTear, M.F. 2002. Spoken dialogue technology: enabling the conversational user interface, *ACM Computing Surveys*, **34**(1):90-169.

Paek, T., and Pierracini, R. 2008. Automating Spoken Dialogue Management design using machine learning: An industry perspective, *Speech Communication*, **50**(8-9):716-729.

Rich, C., and Sidner, C.L. 1998. Collagen: A Collaboration Manager for a Collaborative Interface Agent, *User Modeling and User Assisted Interaction*, **7**(3-4):315-350, Kluwer Academic Publishers.

Traum, D., and Larsson, S. 2003. The Information State Approach to Dialogue Management. *Current and New Directions in Discourse and Dialogue*, Text Speech and Language Technology, **22:**325-353.

Williams, J., and Young, S., 2005. Scaling up POMDPs for Dialog Management: The "Summary POMDP" Method, *ASRU'05.*

Williams, J., and Young, S., 2007. Partially Observable Markov Decisions Processes for Spoken Dialog Systems, *Computer, Speech and Language*, **21**(2).

Williams, J., 2010. Incremental Partition Recombination for Efficient Tracking of Multiple Dialog States, in *Proc. of ICASSP, Dallas, Texas, USA.*

Young, S. 2006. Using POMDPs for Dialog Management, in *Proc. of SLT-2006*, Palm Beach, Aruba.