Timed Petri Nets for Multimodal Interaction Modeling

Crystal Chao
School of Interactive Computing
Georgia Institute of Technology
Atlanta, Georgia, USA
cchao@gatech.edu

Andrea L. Thomaz
School of Interactive Computing
Georgia Institute of Technology
Atlanta, Georgia, USA
athomaz@cc.gatech.edu

ABSTRACT

Humans naturally use the multiple modalities of speech, gesture, and gaze when they communicate. They also engage in turn-taking to manage the seizing and yielding of the speaking floor, a process that controls the execution of turns comprising discrete speech, gesture, and gaze events. We are interested in modeling such interaction processes for a social robot to use that can be transferred between different domains. Timed Petri nets (TPNs) are currently uncommon in human-robot interaction (HRI) but offer an attractive representation for modeling concurrency and synchronization when controlling behavior for multiple modalities. Their representation also permits the intuitive and modular combination of rule-based behavior expression with statistical timing models. We describe their utility in application to multimodal interaction and compare them to finite state machines (FSMs) and Markov models, two more commonly used methods for control.

1. INTRODUCTION

Cooperation between humans is characterized by temporally extended action. Humans perform communicative acts that engage the multiple modalities of speech, gesture, and gaze, while simultaneously applying their bodies to the tasks at hand. Such social exchanges often feature:

- *synchronization* over bottlenecks, as when taking turns with the speaking floor in a dialogue, handling shared objects in the environment, or waiting for the visual attention of a partner;
- *concurrency* of actions across participants, as well as of actions across the modalities of a single participant;
- *sequences* of conditions that must be met, as when following a plan or executing conversation following a situational interaction "script";
- *timing* management, so that the collaboration is efficient and that interaction dynamics meet expectations.

A robot or embodied agent designed to interact with a human in such a cooperative setting requires a behavior system that can model these types of interactions. We believe that timed Petri nets (TPNs) offer a natural representation for multimodal interactions, despite being uncommon in robotics. Although almost any robot architecture can be molded to fit any small-enough problem, TPNs seem to offer some representational advantages for scaling complex implementations and for transferring behavior between domains.

One issue for progressing a robot's HRI capabilities is the problem of creating general "social intelligence" interaction modules that produce transferable behavior across multiple domains. In particular, our research has focused on the development of a general-purpose turn-taking module. This stands in contrast with the idea of discovering transferable *principles* through user studies, to be applied case-by-case in a hard-coded fashion. Our goal is to move away from a model where each new domain requires painstaking setting of each gesture and glance, and towards a model where new domains only specify new semantics and a few behavioral parameters. We offer a systems perspective on the advantages and disadvantages of using TPNs for multimodal interaction modeling as compared to some of the more commonly established representations in robotics.

2. BACKGROUND

Petri nets have been popular in previous decades for workflow modeling due to their rich representation and intuitive graphical notation. Past applications to robot control include domains of assembly or manufacturing, and more recently, supervisors for multi-robot control in a robot soccer domain (e.g. [1]). In HRI, Holroyd has applied them to execution of Behavior Markup Language (BML) [5].

In modern robotics, Markov decision processes [6] and Markov chains are extremely popular representations. Finite state machines (FSMs) are also more commonly used. We think that in most cases where multiple FSMs are parallelized, Petri nets would probably be better suited. We also think that Markov models, which are meant to model uncertainty over sequences, are awkward for modeling temporally extended actions in HRI.

2.1 Timed Petri nets

Timed Petri nets are an extension of Petri nets with additional modeling of timing. A basic Petri net is a bipartite multigraph of alternating places P and transitions T, connected by directed arcs. Tokens are discrete resources that reside in and move between places through the firing of

ICMI 2012 Workshop on Speech and Gesture Production in Virtually and Physically Embodied Conversational Agents, October 26, 2012, Santa Monica, California, USA.

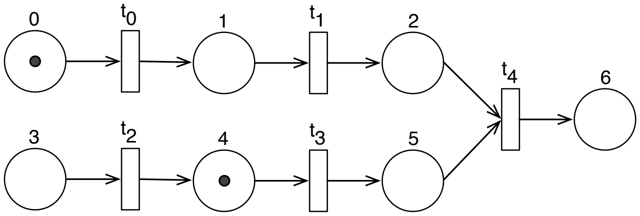

Figure 1: Example Petri net that models sequence, concurrency, and synchronization.

transitions; control is transferred through token movement throughout the graph. Figure 1 shows a graphical example with seven places (0–6), five transitions (t_0–t_4), and two tokens (in places 0 and 4). The mapping of places to tokens is called a marking $M : P \to \mathbb{N}$, and is the Petri net's implicit state representation. The Petri net in Figure 1 is marked [1000100]. More formally, a Petri net is a 5-tuple $N = (P, T, I, O, M_0)$, where:

- P is a finite set of places,
- T is a finite set of transitions, where $P \cup T \neq \emptyset$ and $P \cap T = \emptyset$,
- $I : P \times T$ is the input function directing incoming arcs to transitions from places,
- $O : T \times P$ is the output function directing outgoing arcs from transitions to places, and
- M_0 is an initial marking.

A transition $t = \{\mathcal{G}(I), \mathcal{F}(M, I, O)\}$ is controlled by two functions: a guard function $\mathcal{G}(I) \to \{0, 1\}$ and a firing function $\mathcal{F}(M, I, O) \to M'$. The guard function enables the transition (allows it to fire) as a function of the inputs. The firing function runs until it moves tokens from t's input places to its output places, resulting in the transition disabling. More on Petri nets can be found in [7].

In the timed Petri net extension, these functions are associated with timers for the enabling delay and the firing delay. Restrictions to the time distributions result in different classes of Petri nets, e.g. stochastic Petri nets (exponential), time Petri nets (deterministic intervals), etc. [8].

In Figure 3, we provide an example of modeling multimodal robot behavior using a Petri net. The example shows an action with spatial deixis, such as pointing and gazing at an object while uttering speech referring to that object.

2.2 Finite state machines

One of the most commonly used automata for agent action is the finite state machine. Finite state machines are well suited for modeling simple sequential control. They are formally defined as:

- S is a finite set of states,
- X is an alphabet of input symbols,
- Y is an alphabet of output symbols,
- $T : S \times X \to S$ is a transition function between states,
- $O : S \times X \to Y$ is an output function, and

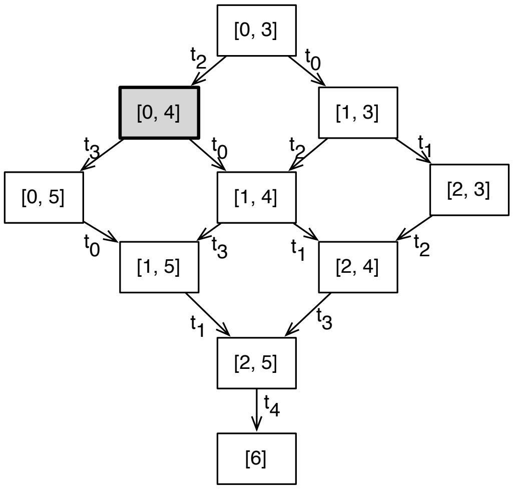

Figure 2: FSM (and reachability graph) corresponding to the Petri net in Figure 1. The FSM states are written as tuples of Petri net places that concurrently contain tokens.

- $s_0 \in S$ is an initial state.

Both Petri nets and FSMs match a development style that focuses on modeling conditional rules. Structurally, any FSM can be represented in Petri net form. Such a Petri net requires transitions to have a maximum of one input place and one output place, and the net contains exactly one token. The key limitation of the FSM is that only one state is active at a time, which is what limits its utility in applications requiring concurrency. It is difficult for systems relying on FSMs to model both synchronization and concurrency. One can parallelize FSMs to represent concurrency, but then synchronization across the FSMs is not modeled explicitly, which results in unpredictable behavior and scalability issues—oft-cited caveats of the subsumption architecture [2]. If one wants to model synchronization instead, the crossproduct of any concurrent conditions must be taken to enumerate the full state space. An example is shown in Figure 2, the FSM for the Petri net in Figure 1; this is also the reachability graph of the Petri net, which connects markings that are successively reachable from each other as a result of transition firings. For those already familiar with FSMs, Petri nets offer a principled way to integrate multiple FSMs.

2.3 Markov chains

A Markov process describes a sequence of states obeying the Markov property, meaning that the probability distribution for transitioning from one state to the next is conditionally independent of those at previous or future time steps. A discrete-time Markov chain is defined as:

- S is a finite or countable set of states,
- $X(n) = X_0, X_1, ...$ is a sequence of S-valued random variables at time steps $n = 0, 1, ...$
- $Pr(X_n = j | X_{n-1} = i)$ for $i, j \in S$ is a transition probability function obeying the Markov property.

Like FSMs, Markov models are state-based representations. Figure 2 can depict a Markov chain with the following

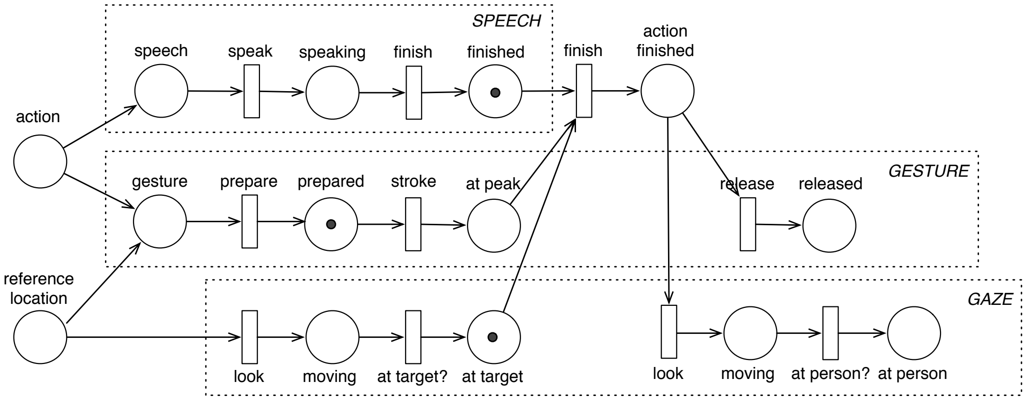

Figure 3: An example of modeling multimodal behavior using a Petri net. This diagram sketches a communicative act with spatial deixis. The agent can utter a spoken phrase (e.g. "It goes over here") while gesturing and gazing towards the reference location. The boxed-in subgraphs belong to the subprocesses of speech, gesture, and gaze; the others form an action-level interface. The nodes with the same name in the gaze process are actually the same nodes in the system; they are duplicated here for temporal understandability.

adjustments: each directed edge is labeled with a probability of transitioning, and each state also has an edge directed at itself for self-transitions. The transition function is applied at each time step to yield a posterior distribution indicating the likelihoods of being in each state. Certain classes of TPNs and Markov chains have equivalent dynamics. For example, stochastic Petri nets have exponential firing times associated with their transitions and thus can be mapped to continuous-time Markov chains.

Other Markov models, like hidden Markov models (HMMs) or partially observable Markov decision processes (POMDPs), have similar characteristics to the Markov chain. In Section 3, we refer frequently to Markov chains, but the representational characteristics being discussed generally apply to all types of Markov models.

3. ISSUES IN MODELING

Here we summarize some of the relevant concerns about developing systems to control robots in HRI settings.

3.1 Scalability

A Markov chain with N states can be represented by an $N \times N$ transition matrix containing transition probabilities between pairs of states, usually trained from data in an unsupervised fashion (or potentially hand-coded, which can be a less than intuitive process). Because all potentially concurrent conditions must be coupled together to form states in Markov chains and FSMs, this poses an issue with the system's scalability. The state space grows exponentially with the number of concurrent conditions because all values possible for each dimension must be combined with all other dimensions' values. This can be problematic for systems that model many modalities.

The enumeration of the minimal set of combinations of concurrent conditions may be unintuitive. One strategy easily enabled by a Markov chain representation is to allow combinations of any conditions' values regardless of the actual relationship between them. The true structure of whether conditions co-occur is reflected in the sparsity of the trained transition matrix. This approach can be beneficial for smaller problems but naturally worsens the model's scalability. A way to deal with the tractability of state-based representations is to expend effort on pruning and merging states after training, a familiar practice for users of POMDPs. In comparison, the number of Petri net nodes scales linearly with additional concurrent conditions (adding additional places), thereby remaining much more manageable and compact in the face of increasing complexity.

3.2 Generalizability

Another issue with state-based representations is that it is currently difficult to generalize the resulting model to new domains (although transfer learning is an active research area). Because concurrent conditions must be coupled together to form states, this necessarily includes domain-specific conditions in all of the states. Any modifications to the state space require that the model be retrained, whether it be changing domains or extending behavior. Also, there's no portion of a Markov model that can be easily extracted and reused; the system stands as a whole or not at all.

In contrast, it is possible for a TPN to be decomposed into multiple subprocesses, each a TPN in itself, that are connected by interfaces. Properly abstracted subprocesses can be reused by connecting them to new domain-specific models. Multiple people could develop separate subprocesses with an agreed-upon interface, as in standard software engineering. The graphical notation facilitates the communication and interpretation of such designs.

TPNs are also easy to modify locally. When new nodes are added, connected transitions must have their firing dynamics specified (i.e. probabilistic firing distributions, interval timers, etc.), but the entire system does not need to be retrained. The combination of the modularity of TPN process design and its relative extensibility makes it an attractive representation for iteratively developing the social cognition of a robot or agent.

3.3 Time Representation

Another drawback of Markov chain-based representations for this application is the indirect representation of time. In Markov chains, time passed in a state is implicitly represented through self-transition probabilities. That is, when a state in a discrete-time Markov chain transitions to itself repeatedly over discrete time steps before it switches to another state, some multiple of the discrete time step passes as a result. The time passed thus follows a geometric distribution. The continuous-time Markov chain describes a variant with time following the exponential distribution, the continuous analogue of the geometric distribution.

In the particular application of modeling the timing of events within a temporally extended action, it is indirect to work in probability space instead of time space. It is also restricting to limit oneself to the memoryless exponential and geometric distributions, which can result in inappropriately timed transitions. Some techniques overcome this by including discretized time as a dimension in the state space. An example is [3], which used a time-indexed POMDP to train an agent's turn-taking within a driving domain. Incorporating time in this way aggravates the scalability issue by exploding the state space and forces a tradeoff between tractability and model expressivity. Again, much effort needs to be expended on pruning or merging states.

In addition, memoryless probabilistic state switching can result in nonsensical behavior, as there is a chance that states will switch unnaturally quickly. Although Markov models can generate summary time characteristics such as expected times or durations, the model does not represent any temporal continuity. In order to use the model reasonably and safely, some additional filtering is likely required, which disrupts the dynamics that the system was trained to model. The TPN allows the setup of temporally coherent actions, where transition firing dynamics can be specified intuitively in terms of the relevant space (time), whether they are derived statistically from data or hand-coded.

3.4 Analyzability

A system's analyzability describes the ease with which one can answer questions about its dynamics. Examples are the percentage of time spent in a state, or the probability of one condition given another. For example, in analyzing human-robot teamwork, it may be important to evaluate the percentage of time the human or robot was idle.

When it comes to formal methods for analysis, Markov chains have the advantage. They have been popular because they are backed by efficient algorithms that can answer questions about uncertainty regarding past and future observations. In fact, Petri nets are often converted to their Markov chain duals for performing analysis. However, this superiority applies specifically to standard memoryless models, i.e. following geometric or exponential distributions. Realistically in HRI, such assumptions don't apply to timing requirements of interactions. Semi-Markov models can be used to model other timing distributions but are much more difficult to analyze because the efficient algorithms, e.g. Baum-Welch, no longer apply.

In our work, we have found that simulation-based methods of analysis are required to be maximally general [4]. Although many runs are needed and results are not as satisfying as from closed-form proofs, the method allows for mixtures of transitions with arbitrary firing timing. It seems to us that significant complexity of naturalistic interactions must be sacrificed in order to model them as Markov chains, with their limited scalability and representational accuracy, for the sake of closed-form analyzability. When the original problem requires so much simplification for the sake of tractability, it actually makes little sense to point to "optimal" policies and inferences in POMDPs and Markov chains. We thus consider that TPNs still offer an attractive alternative over the throwaway system designs fostered by state-based representations.

4. DISCUSSION

Our perspective is that state-based representations such as FSMs and Markov chains can work quite well if one only needs to deliver an interaction within a single domain. The indirectness of the representation of time in the Markov chain can make development more difficult, but there are workarounds possible.

Where representational issues pose more of a concern is in developing general systems that support increased complexity. The literature on HRI contains many principles on how robots should act, as ascertained through user studies featuring single behaviors. Still new principles are being discovered and published all the time. It is less than ideal to have to hand-code such principles into the robot's behavior for each new domain of interaction with no hope for transfer. If such principles could be encapsulated in modular, transferable processes, perhaps better progress could be made towards achieving richer social behavior in interactive robots. Considering the limitations of the prevailing methods for producing robot actions, TPNs seem to offer advantages in scalability, generalizability, and representation of time.

5. CONCLUSION

Multimodal human-robot interactions tend to feature synchronization, concurrency, condition sequences, and timing requirements. Timed Petri nets are well suited for modeling these types of control flows. Due to representational advantages when scaling to more complex systems, they are worth considering as an alternative to the more commonly used methods of FSMs and Markov chains.

6. REFERENCES

[1] L. R. Barrett. *An Architecture for Structured, Concurrent, Real-Time Action.* PhD thesis, University of California, Berkeley, 2010.

[2] R. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14–23, 1986.

[3] F. Broz. *Planning for Human-Robot Interaction: Representing Time and Human Intention.* PhD thesis, Carnegie Mellon University, 2008.

[4] C. Chao and A. Thomaz. Timing in multimodal reciprocal interactions: Control and analysis using timed Petri nets. *Journal of Human-Robot Interaction*, 1(1):46–67, 2012.

[5] A. G. Holroyd. Generating engagement behaviors in human-robot interaction. Master's thesis, Worcester Polytechnic Institute, 2011.

[6] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134, 1998.

[7] T. Murata. Petri nets: Properties, analysis and applications. In *Proceedings of the IEEE*, volume 77, pages 541–580, 1989.

[8] J. Wang. *Timed Petri Nets: Theory and Application.* Springer, 1998.