REINFORCEMENT LEARNING OF DIALOGUE STRATEGIES WITH HIERARCHICAL ABSTRACT MACHINES

Heriberto Cuayáhuitl[1], *Steve Renals*[1], *Oliver Lemon*[2], *Hiroshi Shimodaira*[1]

CSTR[1], HCRC[2], School of Informatics, University of Edinburgh
2 Buccleuch Place, EH8 9LW, Edinburgh, Scotland, UK
{h.cuayahuitl, s.renals, olemon, h.shimodaira}@ed.ac.uk

ABSTRACT

In this paper we propose partially specified dialogue strategies for dialogue strategy optimization, where part of the strategy is specified deterministically and the rest optimized with Reinforcement Learning (RL). To do this we apply RL with Hierarchical Abstract Machines (HAMs). We also propose to build simulated users using HAMs, incorporating a combination of hierarchical deterministic and probabilistic behaviour. We performed experiments using a single-goal flight booking dialogue system, and compare two dialogue strategies (deterministic and optimized) using three types of simulated user (novice, experienced and expert). Our results show that HAMs are promising for both dialogue optimization and simulation, and provide evidence that indeed partially specified dialogue strategies can outperform deterministic ones (on average 4.7 fewer system turns) with faster learning than the traditional RL framework.
Index Terms: reinforcement learning, spoken dialogue systems.

1. INTRODUCTION

A dialogue strategy is a key component for a spoken dialogue system because it governs the control flow of the conversation. Ideally, dialogue strategies should lead dialogue systems towards successful, efficient and natural conversations. However, designing such dialogue strategies is a challenging task without a simple solution. On the one hand, dialogue strategies must collaborate with imperfect components such as the automatic speech recognizer (ASR). On the other, dialogue strategies must consider all possible situations in the conversation taking into account the ASR performance, type of user, database content, etc. Furthermore, whilst designing dialogue strategies for system-initiative and small-scale dialogue systems may be straightforward, the opposite occurs for mixed-initiative and larger-scale dialogue systems. This means that as the dialogue complexity increases, dialogue strategies designed by humans are more prone to errors, labour-intensive and non-portable. These facts motivate the topic of semi-automatic dialogue strategy design.

Previous research efforts have proposed several methods for dialogue strategy design. The most basic methods are based on deterministic finite state machines, where the states represent questions and the transitions control the flow of the conversation [1]. These methods have been successful for system-initiative dialogue systems, but are impractical for mixed-initiative dialogue systems. A more recent approach applies the reinforcement learning framework [2], in which an agent learns the best actions for every situation in the conversation. This approach is a promising solution for mixed-initiative dialogue systems; however, it is computationally expensive since it requires many dialogues to learn near-optimal dialogue strategies. Potential solutions for this problem include reduced search spaces (before learning) [2,3], function approximation [4] and dialogue simulation [5,6]. However, there is a lack of a principled methodology for reducing search spaces to manageable sizes.

Research mainly sponsored by PROMEP (http://promep.sep.gob.mx) and the Autonomous University of Tlaxcala (www.uatx.mx), and partially sponsored by the EU-FP6 project "TALK" (www.talk-project.org).

In this paper we propose to design dialogue strategies using a combination of Finite State Machines (FSMs) and Reinforcement Learning (RL) in the context of Markov Decision Processes (MDPs). The basic idea is to design dialogue strategies using FSMs, specifying obvious actions deterministically and specifying difficult actions stochastically; the latter are the ones to be optimized. For such purpose we apply RL with Hierarchical Abstract Machines (HAMs) [7,8]. This method offers the following benefits among others: a) partially specified dialogue strategies, because the system developer decides what to hand-craft and what to optimize; b) faster learning, because the RL agent uses reduced search spaces due to the prior knowledge incorporated in the HAMs; c) potentially improved performance, because RL optimizes parts of the dialogue strategy; and d) knowledge transfer, because the HAMs may be reusable. In addition, we propose to build simulated user models using HAMs, incorporating hierarchical deterministic and probabilistic behaviour.

In this paper we assume the reader is familiar with the fundamentals of FSMs, MDPs and RL.

2. REINFORCEMENT LEARNING WITH HIERARCHICAL ABSTRACT MACHINES

A Hierarchical Abstract Machine (HAM) is a program that constrains the actions that an RL agent can take in each state [7,8]. HAMs are similar to non-deterministic FSMs whose transitions may invoke lower-level machines. When a hierarchical machine is called, control is transferred to the start state, where machine states are visited until reaching a stop state, which returns control to the caller, and then determines the next machine state, and so on until reaching the stop state of the root machine.

A HAM is a collection of three-tuples $H_i = (\mu, I, \delta)$, where μ is a finite set of machine states, I is the initial state, and δ is the transition function determining the next state using either deterministic or stochastic transitions. The main types of machine states are: *start* (execute the current machine), *action* (execute an action), *call* (execute another machine), *choice* (select the next machine state), and *stop* (halt execution and return control). A machine H_i is *abstract* (or partially specified) if it specifies non-deterministic choice states.

For any MDP M and any HAM H, there exists an induced MDP $M' = H \circ M$ [7]. The solution defines an optimal policy that maximizes the expected total reward by an RL agent executing H in M.

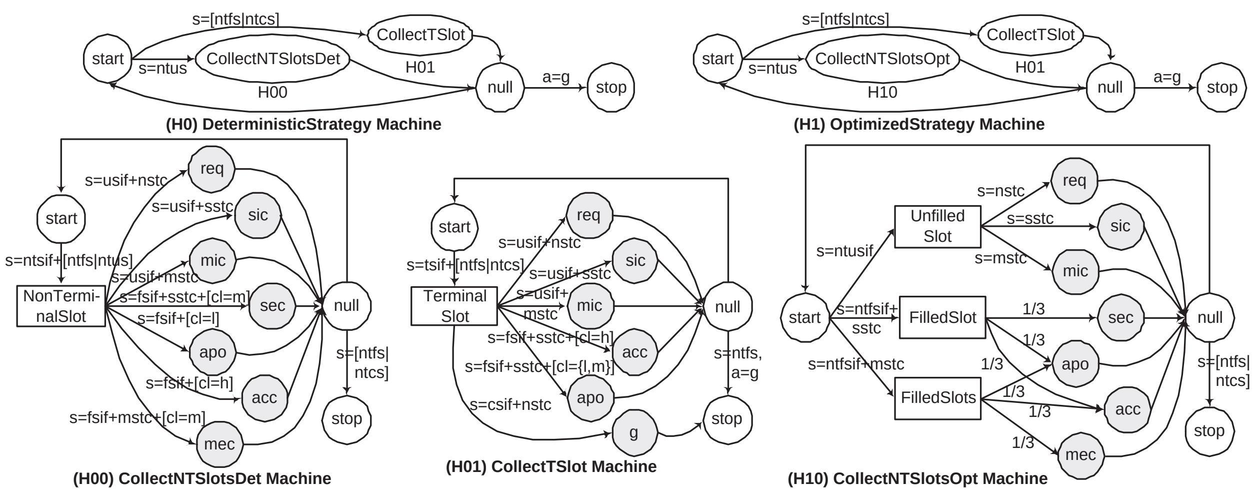

Fig. 1. *Hierarchical deterministic and abstract machines for two generic dialogue strategies: Deterministic (left) and Optimized (right). Notation: Ellipses=call states, rectangles=choice states, lightly shaded circles=action states.*

A brief description of the induced MDP $M' =< S', A', T', R' >$ is as follows: a) the set of states S' is the cross-product[1] between the choice states of H and the states of M, b) the set of actions A' for a given state corresponds to the action states (or call states) that change only the machine component, c) the transition function T' corresponds to executing the transition functions T and δ in parallel, and d) the reward function R' is the same as R for single-step actions, otherwise the reward is zero.

The aim of the induced MDP is to work with a reduced search space using single-step and multi-step (or high-level) actions, the latter correspond to call states. As a consequence, the induced MDP is in fact a Semi-Markov Decision Process (SMDP), because actions can take more that one time-step to complete.

A learning algorithm for the induced SMDP is a variation of Q-learning called SMDP Q-learning[2]. This algorithm can be applied to the HAMs framework using an extended Q-table $Q([s,m],a)$, which is indexed by an environment state s, machine state m, and action a taken at a choice state m. In this way, the algorithm applies the following update rule from choice state to choice state:

$$Q([s,m],a) \leftarrow Q([s,m],a) + \\ \alpha[r + \gamma^{\tau} \max_{a'} Q([s',m'],a')] - Q([s,m],a)],$$

where $r = r_{t+1} + \gamma r_{t+2} + ... + \gamma^{\tau-1} r_{t+\tau}$, and τ is the number of time steps elapsed between state s and state s'.

3. DIALOGUE STRATEGY OPTIMIZATION

3.1. Designing Partially Specified Dialogue Strategies

The idea of partially specified dialogue strategies serves two important purposes. First, to give freedom to the system developer in what to specify manually and what to optimize; and second, to reduce search spaces due to the fact that they grow exponentially using the

[1]Parr and Russell [7] propose a method to reduce large induced MDPs, but we used the following conditions: a) parent transitions of choice states are taken into account, b) environment states that did not match any choice state are removed, and c) states with an empty set of actions are also removed.

[2]SMDP Q-Learning converges under similar conditions as Q-Learning.

Table 1. *State-action space representation*[4−6].

STATE (dialogue history , slot in focus)	ACTIONS
q0.u|q1.u|q2.u|q3.u|q4.u,q0	req,apo,sec,sic,mec,mic,acc
q0.l|q1.u|q2.u|q3.u|q4.u,q1	req,apo,sec,sic,mec,mic,acc
q0.m|q1.u|q2.u|q3.u|q4.u,q2	req,apo,sec,sic,mec,mic,acc
q0.h|q1.u|q2.u|q3.u|q4.u,q3	req,apo,sec,sic,mec,mic,acc
...	...
q0.c|q1.c|q2.c|q3.c|q4.c,q4	g

standard RL framework. We propose the following methodology to design optimized dialogue strategies: 1) Design an MDP by choosing an appropriate representation of states, actions and reward function; 2) design a dialogue strategy using HAMs, 3) generate the induced (S)MDP $M' = H \circ M$, 4) learn a dialogue strategy using M' and simulated users[3], and 5) test the learnt dialogue strategy.

As an illustrative case study, consider a single-goal flight booking dialogue system with slots[4] $Q = \{q0, q1, q2, q3, q4\}$, state variable describing the slot status[5] $V = \{u, l, m, h, c\}$ and actions[6] $A = \{req, apo, sec, sic, mec, mic, acc\}$. Assume that the environment states are compounded by dialogue history and slot in focus q_i, where the former has combinations of Q and V separated with the character "|" (see table 1). The size of this state space is computed as $|S| = |V|^{|Q|} \times |Q|$ plus the terminal state. Thus, the size of the full state-action space corresponds to $|S \times A| = 109376$ state-actions.

The rest of this section focuses on step two of our methodology, which describes the design of two dialogue strategies using HAMs,

[3]Learning dialogue strategies with real users is possible but impractical.

[4]Slots: q0=departure city, q1=destination city, q2=departure date, q3=departure time, q4=flight offer. The last slot within a dialogue goal is referred to as the terminal slot, the others are non-terminal.

[5]Values of the state variable slot status: u=unknown, l=low confidence, m=medium confidence, h=high confidence, c=confirmed.

[6]Actions: req=request, apo=apology, sec=single explicit confirmation, sic=single implicit confirmation (with request), mec=multiple explicit confirmation, mic=multiple implicit confirmation (with request), acc=accept slot in focus (and move to the next unfilled slot, from left to right), g=goal.

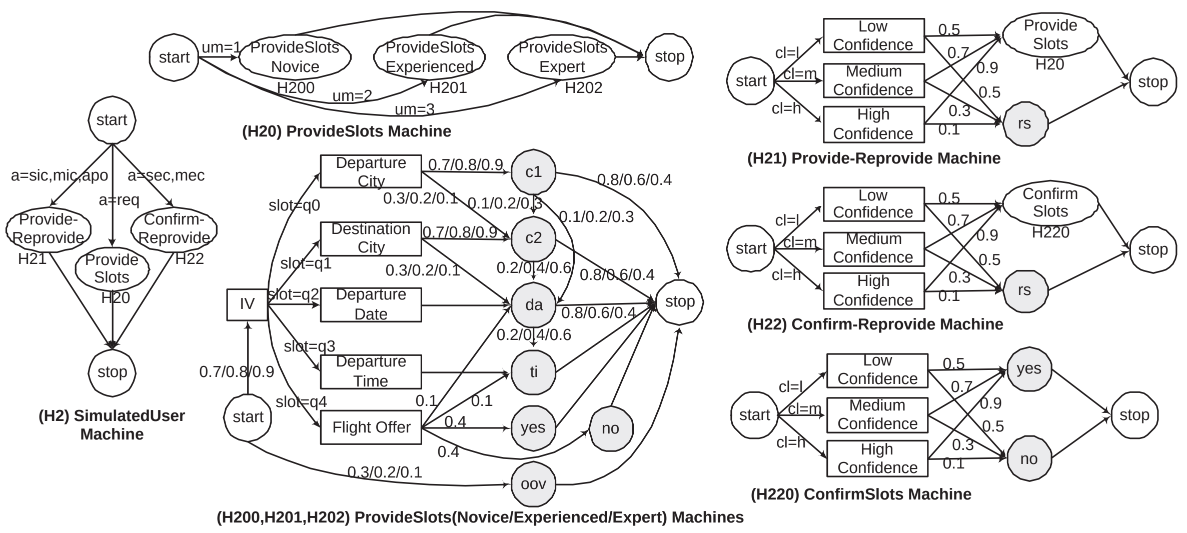

Fig. 2. *A simulated user model using hierarchical abstract machines, where sequences of action states correspond to user responses.*

illustrated in figure 1. The first strategy uses a deterministic HAM, labelled as H0[7]. Notice that this machine has only deterministic transitions, meaning that no optimization is required at all. But if we perform the cross product of this machine and the state space of our case study, the size of the induced state-action space is $|S' \times A'| = 2261$, representing only 2.06% of the full search space. However, there may be better dialogue strategies than deterministic ones, and reinforcement learning with HAMs aim to find those ones.

The second strategy for our case study uses a HAM with deterministic and stochastic transitions, labelled as machine H1. It means that whilst deterministic transitions of choice states correspond to one action per environment state, stochastic transitions produce a reduced action set that will be optimized using RL. For instance, the action set for environment state $s = q0.m|q1.u|q2.u|q3.u|q4.u$, $q0$ is $a = \{sec, apo, acc\}$, because the slot in focus q_0 is filled and there is a single slot to confirm, corresponding to the transition condition “s=ntfsif+sstc”. The cross product of the machine H1 and the state space of our case study yields a state-action space of size $|S' \times A'| = 5261$, representing only 4.81% of the full search space. Obviously, the quality of the learnt dialogue strategy will depend on the constraints specified in the HAM, but there are benefits at the same time: a) tailored dialogue optimization, b) faster learning, c) reduced computational demands, and d) reusable HAMs.

3.2. User Simulation Using HAMs

Most of the previous work in user simulation for optimizing dialogue strategies uses statistical techniques [5]. Such methods are useful because they explore a vast amount of user behaviour due to the randomness in the models, but they can yield incoherent user responses. In this paper we propose hierarchical behaviour modelling for user simulation, and use HAMs such a purpose. HAM-based user models are attractive for the following reasons: a) models can be fully-handcrafted, this is useful in the absence of training data; b) models can be fully learnt from data, this is useful in the presence of training data; c) models can be partially specified, this is useful because the combination of hierarchical deterministic and probabilistic behaviour may yield more coherent user responses; and d) HAM-based user models assume that they know the current state and action of the environment, which may yield more consistent responses. This makes HAM-based user models different from previous approaches.

Figure 2 illustrates a hand-crafted HAM-based simulated user model using intentions[8], suitable to interact with the spoken dialogue system described in the previous section. This model includes three types of user: novice, experienced and expert. The execution of the HAM H2 generates a user response given by the sequence of visited action states[9]. As an example, consider the environment state $s = q0.u|q1.u|q2.u|q3.u|q4.u$, $q0$, action $a = req$, and type of user $um = 1$. The machine H2 invokes the child machine H20 (because the action is a request), then it invokes the machine H200 (because the user type is novice), then it chooses between an in-vocabulary or out-of-vocabulary response, the former takes into account the slot in focus to observe action states, and so on until the stop state of the root machine is found. Notice that machines H200-H202 (compactly illustrated) have different probabilities for each type of user. These machines model user behaviour according to the following assumptions: a) novice users behave with more confusion and less initiative, b) expert users behave with less confusion and more initiative, c) experienced users behave between novice and experts, and d) positive confirmations are more likely to higher confidence levels.

The justification for building a hand-crafted simulated user model is due to the lack of richly annotated dialogue corpora for training models. For instance, the DARPA Communicator corpora does not include annotations for ASR confidence levels. Nevertheless, the proposed user model utilizes empirical knowledge to model simple reasonable behaviour of real users. Furthermore, the use of more complex hierarchies could describe more complex behaviour, and they may be reusable and/or task independent, facilitating the optimization of dialogue strategies in new domains.

[7]Abbreviations: s=state, a=action, ntus=non-terminal unfilled slots, ntfs=non-terminal filled slots, ntcs=non-terminal confirmed slots, ntsif=non-terminal slot in focus, tsif=terminal slot in focus, usif=unfilled slot in focus, fsif=filled slot in focus, csif=confirmed slot in focus, nstc=none slots to confirm, sstc=single slot to confirm, mstc=multiple slots to confirm, cl=confidence level, ntusif/ntfsif=non-terminal unfilled/filled slot in focus.

[8]Level of communication above words, analogous to dialogue acts.

[9]Intentions: yes, no, c1=departure city, c2=destination city, da=departure date, ti=departure time, oov=out-of-vocabulary response, rs=reprovide slots.

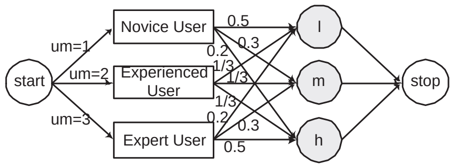

Fig. 3. *The confidence level model using an abstract machine.*

4. EXPERIMENTS AND RESULTS

4.1. Experimental Setup

Our experiments used a single-goal flight booking spoken dialogue system, as described in section 3.1. The aim was to investigate the performance of deterministic and optimized dialogue strategies using different types of user, by applying the proposed approach of partially specified dialogue strategies. In our experiments only the optimized dialogue strategy required learning. These experiments utilized a simulated environment including a simulated user model and an ASR confidence level model. Whilst the simulated user model was described in section 3.2, the ASR confidence level model used the HAM illustrated in figure 3. This HAM observes confidence levels given the type of user, and its stochastic transitions apply the following assumption: ASR performance is better (i.e., higher confidence levels) for expert users than for novice users. Thus, the induced MDP was generated as follows: a) states and actions as described in sections 2 and 3.1, but we only used single step actions, and leave multi-step actions as future work; b) the transition function used deterministic transitions, based on the current environment-machine state, action, user response and confidence level; and c) the reward function evaluated dialogue length, consisting of $+100$ if all slots were confirmed and 0 otherwise.

Our experiments used the following learning setup: algorithm $=$ SMDP Q-Learning, equivalent to Q-Learning when using single time steps; step size $\alpha = 100/(100+t)$, with t elapsed time-steps; discount factor $\gamma = 0.9$; selection strategy $= \epsilon$-greedy, with 20% exploration; initial Q-values$= 0$; and learning episodes$= 10^5$.

4.2. Results

Figure 4 shows test results comparing both dialogue strategies (deterministic and optimized) using three types of user (novice, experienced, and expert). The X-axis corresponds to the type of user used to test the dialogue strategies, and the Y-axis corresponds to average number of system turns. All dialogues were successful, the only difference being in the number of turns taken to reach the final state. The optimized strategies report a cross evaluation using the three types of user and a mixture of them, the last (ALL) means that each episode (dialogue) used a randomly chosen user.

Each bar reports the mean and standard deviation (thin lines), and used 10^5 simulated dialogues. Our results are statistically significant because of the large number of simulations ($p < 0.01$). We can observe that the optimized strategies performed significantly better for novice users, slightly better for experienced users, about the same for expert users, and significantly better using all users (on average 4.7 fewer system turns than the deterministic strategy). This result tells us that our deterministic strategy was more appropriate for expert users than for the other users. Also, this result tells us that simulated user models must take into account different types of user, otherwise the learnt strategies will be sub-optimal for different users.

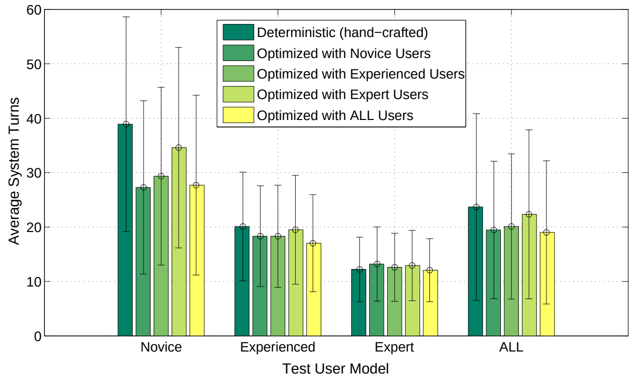

Fig. 4. *Test results of dialogue strategies on different user models.*

5. CONCLUSIONS AND FUTURE WORK

In this paper we proposed partially specified dialogue strategies for dialogue strategy optimization. For this purpose we used Reinforcement Learning (RL) with Hierarchical Abstract Machines (HAMs). We also proposed to build simulated user models using HAMs. Our findings are as follows: a) HAMs are useful to reduce search spaces for dialogue optimization, b) HAMs are useful for modelling user behaviour in a hierarchical way, c) user simulation must take into account different user types, and d) partially specified dialogue strategies are promising due to the fact that they can outperform deterministic ones (on average 4.7 fewer system turns using the optimized strategy with all users) with fewer computational demands than standard RL (less than 5% of the full search space in our case study).

Recommended future work is as follows: a) more complex and larger-scale spoken dialogue systems, b) hierarchical reinforcement learning of dialogue strategies using SMDPs and partially observable SMDPs, c) comparison with function approximation methods e) evaluation of task-(in)dependent/hand-crafted/learnt HAM-based user models, and f) experiments with real users.

6. REFERENCES

[1] M. McTear. “Modelling Spoken Dialogues with State Transition Diagrams: Experiences with the CSLU Toolkit,” in *Proc. of ICSLP*, pp. 1223-1226, 1998.

[2] S. Singh, D. Litman, M. Kearns, and M. Walker. “Optimizing Dialogue Management with Reinforcement Learning: Experiments with the NJ-Fun System,” in *Journal of AI Research*, 16, pp.105-133, 2002.

[3] H. Cuayáhuitl, S. Renals, O. Lemon, and H. Shimodaira. “Learning Multi-Goal Dialogue Strategies Using Reinforcement Learning with Reduced State-Action Spaces,” in *Proc. of Interspeech-ICSLP*, 2006.

[4] J. Henderson, O. Lemon, and K. Georgila. “Hybrid Reinforcement/Supervised Learning for Dialogue Policies from COMMUNICATOR Data,” in *Proc. of Workshop on K&R in PDS (IJCAI)*, 2005.

[5] J. Schatzmann, K. Weilhammer, M. N. Stuttle, and S. Young. “A Survey of Statistical User Simulation Techniques for Reinforcement-Learning of Dialogue Management Strategies,” in *Knowledge Engineering Review*, Cambridge University Press, 21, pp. 97-126, 2006.

[6] H. Cuayáhuitl, S. Renals, O. Lemon, and H. Shimodaira. “Human-Computer Dialogue Simulation Using Hidden Markov Models,” in *Proc. of ASRU*, pp. 290-295, 2005.

[7] R. Parr, and S. Russell. “Reinforcement Learning with Hierarchies of Machines,” in *Proc. of NIPS*, pp. 1043-1049, 1998.

[8] R. Parr, “*Hierarchical Control and Learning for Markov Decision Processes*,” PhD Thesis, University of California at Berkeley, 1998.