

# Learning Turn, Attention, and Utterance Decisions in a Negotiative Slot-Filling Domain

Ethan O. Selfridge and Peter A. Heeman

*Technical Report CSLU-11-005*

*October 2011*

*Center for Spoken Language Understanding*

*Oregon Health & Science University*

*20000 NW Walker Rd., Beaverton, OR, 97006*

selfridg@ohsu.edu

heemanp@ohsu.edu

**Abstract**—Mixed-Initiative dialogue systems must be effective and natural, and both turn-taking and attention play an important role in meeting these goals. We present the *Tau* Architecture which separates Turn, Attention, and Utterance decisions, and uses Reinforcement Learning to jointly optimize them. The development of sophisticated dialogue managers using Reinforcement Learning requires a simulation domain before any human evaluation, and we describe the Negotiative Slot-Filling domain. This domain is a closer approximation to true mixed-initiative dialogue than any previously used to train dialogue managers. We then detail the *Tau* implementation in the domain, and demonstrate both.

## I. INTRODUCTION

With the advent of mobile devices and relatively accurate speech recognition results, Spoken Dialogue Systems (SDS) have been gaining popularity in recent years. Many current systems are either user-initiative (User: “search coffee shops in downtown portland”) or system-initiative (System: “Where are you leaving from?”). However, as people grow more accustomed to these systems and desire to use them for more complex tasks, the interaction will move more towards mixed-initiative frameworks which involve contributions of both participants to reach the goal (User: “search coffee shops in downtown portland”, System: “Are you interested in the best or the closest coffee shop?”). It is vital that these dialogue systems be effective and natural.

One of the more important aspects of mixed-initiative dialogue systems is turn-taking. In recent years, turn-taking has continued to be of great interest in the computational linguistics and speech recognition communities. Researchers have investigated turn-taking signals [1], backchannel entrainment [2], and multi-party turn-taking [3]. Others have focused on incremental speech recognition which allows the system to react before the user is finished (e.g. [4]). All of this research is, directly or indirectly, working towards effective and natural SDS turn-taking.

Conventional approaches to SDS turn-taking emphasize smoothness [5] and wait for the user to finish before speaking (e.g. [6], [7]). However, there may be times when the system should begin speaking before some indication of user completion. In prior work, using a simple negotiation task, we pro-

posed an SDS which could “bid for the turn” and actively try to take it [8]. This turn-bidding system had substantially shorter dialogues than an SDS which used conventional turn-taking techniques, demonstrating that effective SDS turn-taking in an integral aspect of efficient mixed-initiative dialogue.

Attention is another important aspect of mixed-initiative SDS. Attention refers to the topic of the system’s speech and ideally should be selected to be the optimal contribution to task completion. However, many systems have a predetermined slot sequence [9], while others rely on the User to identify an attention or “subtask” [10]. Georgila et al. [11] used a Reinforcement Learning approach, comparing the success of a system whose reward function included slot order and one that did not. The latter system optimized its attention on purely task success and significant gains were found among a certain population of users.

Our first contribution towards mixed-initiative dialogue agents is an Information State (IS) dialogue management architecture — *Tau* — which separates Turn, Attention, and Utterance decisions. Reinforcement Learning (RL) is used for action selection, and the novelty here is the use of RL to jointly optimize *all* three actions. However, a major drawback to RL is the high number of training episodes which necessitates simulation prior to human user evaluation and has resulted in developing systems in overly simple domains. Our second contribution, a simulated *Negotiative Slot-Filling* (NSF) domain for RL dialogue management development, maintains the simplicity of slot-filling with rich and complex negotiation content. We believe it is a closer approximation of mixed-initiative interaction than previously used for RL dialogue management. After describing the *Tau* implementation in NSF, we demonstrate the utility of the NSF domain by comparing different *Tau* dialogue policies.

## II. BACKGROUND

The *Tau* architecture is based on the Information State Update (ISU) approach [12] which we describe below. Heeman [13] showed how to combine the expressive power of ISU with the decision making power of RL, using preconditions to propose applicable actions and using RL to choose the

ideal one. We utilize this method, and so we also provide background on RL as well.

### A. Information State Update

The ISU approach has been used heavily for dialogue management design [12]. It involves executing a series of rule sets that update an “information state” (IS): the agent’s current view of itself and the environment. The IS not only executes rule sets based on environmental changes (e.g. an utterance) but has internal modifications as well (e.g. add a value to the agenda).

Each rule set has a number of update rules, each with a set of preconditions and effects that are constructed using IS variables. When the rule set is tested the preconditions of each rule are checked and if met, the effects are applied. By using a number of different rule sets applied in certain orders, a developer can easily design an agent with relatively complex behavior. For example, the GoDis dialogue system has a 6 types of rules, including accommodation rules, which enable the system to handle unexpected input [12].

### B. Reinforcement Learning

Reinforcement Learning (RL) has been extremely popular in the last decade for research level spoken dialogue systems [14]. It has conventionally taken the form of utterance selection, but recently turn-taking [7], [8] and attention decisions [11] have also made use of RL.

For an SDS, RL learns an optimal a mapping between a state  $s$  and action  $a$ , where performing  $a$  in  $s$  leads to the highest expected reward (or lowest cost) for the dialogue [15]. An  $\epsilon$ -greedy search is used to estimate Q-scores, the expected reward of some state–action pair, where the agent chooses a random action with  $\epsilon$  probability and the  $\text{argmax}_a Q(s, a)$  action with  $1-\epsilon$  probability. The state space should be formulated as a Markov Decision Process (MDP) for Q-learning to update Q-scores properly. An MDP relies on a first-order Markov assumption in that the transition and reward probability from some  $s_t, a_t$  pair is completely contained by that pair and is unaffected by the history  $s_{t-1}a_{t-1}, s_{t-2}a_{t-2}, \dots$ . For this assumption to be met, care is required when designing the RL state.

## III. THE TAU ARCHITECTURE

In this paper, we introduce the Tau architecture. At its core, Tau separates TURN, ATTENTION, and UTTERANCE actions. Tau uses TURN rules to decide when to speak, ATTENTION rules to decide what to focus on (e.g. a specific slot), and UTTERANCE rules to decide what to say (e.g. “What kind of [attention] do you want?”). Each of these decisions is made by separate rule sets which provide *rule set independence*. For instance, the Single-Utterance turn-taking approach, where a “ping-pong” dialogue occurs, can easily be exchanged for the more sophisticated turn-bidding approach where the SDS can actively try to get the turn. This has no effect on the ATTENTION or UTTERANCE rule sets. Similarly, UTTERANCE

rule sets can be altered without changing ATTENTION structures. This compartmentalized design makes for quick and easy implementation, as well as the ability to optimize one decision while holding the other constant. This is a hallmark of good software engineering as well.

Tau has 5 rule sets: 3 action rule sets, an UNDERSTANDING rule set which updates the IS based on the environment, and a DELIBERATION rule set which updates the IS based on other IS variables. At some time (e.g. after a User utterance or after a partial phrase result), the rule sets are applied in a certain sequence: DELIBERATION, TURN, ATTENTION (depends on TURN), UTTERANCE (depends on TURN), and UNDERSTANDING.

Though the action selection *could* be determined by ‘hand-crafting’ complicated preconditions, we prefer to use Reinforcement Learning for all three actions. This results in three types of RL States. The variables in each state can vary under two conditions: (1) each state contains a tri-ary variable which specifies the decision type, and (2) the MDP is maintained. Rather than try to customize the RL variables for each type of decision, which could result in an MDP violation, we (in this paper) choose to use the same variables for each state type to ensure proper learning. One essential element in using RL with Tau is that the Q-Scores for ATTENTION and UTTERANCE decisions are not updated if the System does not speak. Ensuring this is trivial except for the situation where the System plans to speak (and so must decide what to say) but the User speaks instead. In this case, any post-TURN states and decisions *must* be removed from the MDP for proper learning to occur. Figure 1 provides the simple transition schematic between state types. Though it may seem that the System is first deciding whether to speak and then deciding what to say, RL allows it to implicitly decide what to say as it decides to speak. However, since proper learning relies on the “to speak or not to speak” decision, the TURN rule set is put first.

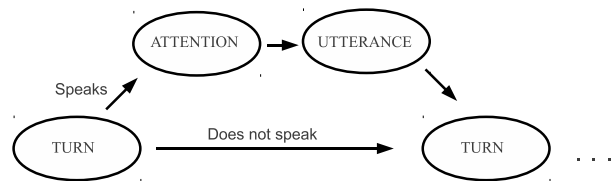


Fig. 1. Simple State to State transition schematic. Notice that TURN only transitions to ATTENTION if the System speaks.

We use only one MDP to learn all three actions, though one can theoretically model an MDP for each action type since we have rule set independence. This would enable state size reduction by maintaining the MDP while tailoring the state for each action. However, this delves into multi-agent RL coordination challenges [16] which is beyond the scope of this contribution and so we leave it for future work.

#### IV. NEGOTIATIVE SLOT-FILLING

The use of Reinforcement Learning lessens the development strain considerably by allowing the agent to discover the best action for a given situation. However, this discovery requires a high number of training episodes and so is not feasible to perform with actual users during research and development. Therefore, the use of simulations have become prevalent: the simulation is either induced from corpora or from some general dialogue situation. Negotiation domains have been sparsely used, the furniture domain being one of the few exceptions (e.g. [17]). However, many negotiation domains model neither natural user turn-taking, nor many common negotiation dialogue acts (such as those described in Sidner [18]) that will be common in true-mixed initiative interactions. If we wish to use RL to develop sophisticated dialogue management techniques, than a richer domain must be used. More importantly, before an evaluation with real users can be done, the design and architecture of the dialogue manager must be relatively complete. It is for this development that these simulations are critical.

To facilitate the development of complex dialogue managers using Reinforcement Learning, we introduce a rich and realistic *Negotiative Slot-Filling* domain (NSF), building and improving on the domain of our previous work [8]. One agent, the System, knows which values are valid fillers for each of three slots. The other agent, the User, has a number of values sorted by preference for each slot. The User has varying beliefs about the validity of each value. Both agents must share information with the other in order to fill all three slots with valid values in a timely manner.

We situate NSF in a food ordering environment. The System is responsible for taking the User’s order for a burger, side, and drink. The System will only accept values that are available, and the User will only accept values with the highest preference. Since the purpose is dialogue management design and not natural language understanding, we use highly stylized semantic speech acts. Both agents have a variety of speech acts that they can perform. They can both *propose* and *reject* values. They can also explicitly *accept* values that have been proposed. The System can give information (*giveInfo*) concerning a value’s availability, indicate when it has given all the information it can on a particular slot (*giveInfoLast*), and *query* the User as to its desired slot value after all the available values have been informed (by *giveInfo* or *propose*). The User can *assert* a value for a slot, and also *retract* a value that it has proposed or *cancel* a value it believes to be accepted. *Assert*, unlike *propose*, does not require an *accept* for it to be accepted by the System. We now further discuss two psycholinguistic phenomena which we model: implicit acceptance and rejection [19], and retraction/cancellation [18]. These are modeled to increase the complexity of the domain to better approximate mixed-initiate dialogue.

##### A. Implicit Acceptance and Rejection

With NSF, we model both explicit and implicit acceptance and rejection. The explicit actions are covered by the *accept*

and *reject* utterances of both agents. In our work, implicit acceptance and rejection is performed according to the rules in the list below. The rules are based on the “Collaborative Principle” [19]: “Conversants must provide evidence of a detected discrepancy in belief as soon as possible [p.4]”. This principle relies on an “assumption shared between conversants” — the ATTITUDE LOCUS — which is the “sequential position in conversation, just after A’s assertion or proposal, in which B first has an opportunity to express B’s attitude or evaluation of A’s utterance [p.4]”. This is modeled in Rules 1 and 2. Towards a more concrete instantiation of implicit rejection, Rules 2 and 3 model the concept of counter-proposal using a “Conflicting Intention [p.20]” form of rejection.

- 1) If the *first* System utterance following a User’s *assert* is a *giveInfo* of the same slot than it is considered rejected by both. Otherwise, it is considered accepted by the User unless Rule 2 takes effect.
- 2) If the System *proposes* at any time following the User’s *assert* or *propose*, it is considered rejected by both.
- 3) If the User *proposes* or *asserts* at any time following the System’s *proposal*, it is considered rejected by both.

##### B. Retraction and Cancellation

We wanted to capture the situation where the User changes their mind mid-order. In NSF, slots are related, and if a certain value is not valid than the User may no longer prefer it. In other words, the User’s preferences change based on what is available. *Cancellations* and *retractions* occur when one value has been proposed or accepted and subsequent value is then rejected. The User only has one combination of values that is achievable, so these speech acts are relatively common.

##### C. User Types and Beliefs

The User is built using ISU (but not Tau), and its behavior is determined by its goal and agenda. Each User begins the dialogue with an arbitrary length Goal Stack where each element has a value for each slot (3 values for each element). The top of the Goal Stack is popped, and the User works to fill slots using that goal. If the System *proposes* a value not in that goal than the User is obligated to *reject* it. Most of the User’s behavior is derived from stochastic processes.

To produce a sophisticated simulated User that should better mimic human users, we use the NSF domain with three different types of users: Novice, Intermediate, and Expert. User’s differ substantially over three different beliefs of value availability: Strong, Warranted, and Weak [20]. This results in different behavior of the User. For example, when working on a value with a Weak belief the User will *propose* instead of *assert*, and the converse is true for a Strong belief. A Warranted belief results in random action selection. Beliefs also change during the dialogue. If the System *proposes* or *givesInfo* on a value than the User’s belief of that value becomes Strong. Beliefs are initialized so that the Expert is more likely to have a Strong belief of an available value, and a Weak belief of an unavailable one. Intermediate users are more likely to have Warranted beliefs for available values, and

TABLE I  
NSF DIALOGUE FRAGMENT WITH EXPERT USER

Line	Speaker	Utterance	Rule
1	User:	assert drink coke	1,2
2	User:	assert burger cheese	
3	System:	propose burger bacon	3
4	User:	cancel drink coke	
5	System:	propose drink Fanta	3
6	User:	propose drink sprite	
...			

Novice's are more likely to have Weak beliefs without regard to availability. Only one combination of desired values is achievable, and Experts will have this achievable combinations within the first half of all its stack of possible combinations. The achievable combination is randomly placed in the Goal Stack for the other two user types.

A sample dialogue is shown in Table 1. Rules governing implicit acceptance and rejection are noted in the final column. An Expert *asserts* two values. The User is then forced to *cancel* (line 4) the inferred accepted drink value after the System implicitly rejects the burger by a counter-proposal (line 3). The System implicitly rejects the Coke with another counter-proposal (line 5). The User then rejects the System's proposal (since it is not the top value the User wants), and proposes its most preferred one (line 6). Turn-bidding (TB) forces both agents to bid after every utterance, and is not shown for clarity purposes. The User bases its bid on the belief of the value in focus, and the System has a variety of bids it chooses from. The stronger the User's belief, the higher its bid will likely be (*Reject*, *Accept*, *Retract* and *Cancel* are all given a Strong belief from which to base the bid). When the User has nothing left to say, it does its bid is effectively 0. We conceptualize bids as utterance onset: The higher the bid, the quicker the utterance.

## V. TAU NSF IMPLEMENTATION

We implemented Tau in the NSF domain: TauNSF. Though the overall architecture is relatively general and not problem specific, the Rule Sets and RL State are domain dependent, and brief descriptions of both are provided below.

### A. ISU Rule Sets

TauNSF has 6 TURN rules: *silence*, *lowest*, *low*, *mid*, *high*, and *highest*. As mentioned previously, we conceptualize bids as utterance onset. Each of the System's bids indicate a range from which a uniformly random number is drawn which indicates the presumed onset. If the System's turn-bid is higher, and so has a shorter onset, than the User's than the System applies its ATTENTION and UTTERANCE rule sets. In a real system it be impossible to win the turn without actually speaking but, as mentioned earlier in Section III, because the subsequent ATTENTION and UTTERANCE decisions would be forgotten in the event that the System does not speak, it is functionally equivalent.

There are 4 ATTENTION rules, 1 for each slot and 1 for when all slots have been filled. These rules assign the variable *attention*, which is used by the UTTERANCE decision.

There are 7 UTTERANCE rules: *query*, *propose*, *giveInfo*, *giveInfoLast*, *reject*, *accept*, and *bye*. Each of them are not available at all times but rather constrained by preconditions. For example, *reject* and *accept* are only available if the User has proposed something

The TauNSF UNDERSTANDING rule set consists of one rule for every potential User or System utterance, as well as others that are responsible for implementing the implicit acceptance and rejection. There is also an UNDERSTANDING rule which assigns the *attention* variable to be the topic of the utterance. It follows that TURN and ATTENTION rules use the *attention* from the previous utterance. TauNSF only has one DELIBERATION rule, which initializes various IS variables.

### B. RL Details

The RL task here is to minimize the number of utterances from the System and User. This is done by giving each utterance the cost 1. Most RL objective functions contain some completion cost which measures the success of the dialogue. However, we do not include this cost since the System will learn to never exit until the task is completed and it will just be measuring utterances anyways. Rather, to speed learning, we use preconditions to ensure the System will not exit until the task is complete.

We now present the RL state that is used to learn the policy for the Tau actions. The RL state keeps track of what kind of decision it is making, and contains the same information for each of the slots. This means that the RL state will be keeping track of information regarding Slot 1 when it is possibly working on Slot 2. For *each* of the slots the following 10 pieces of information are maintained, and combined into one state. All are binary, unless explicitly stated otherwise.

- attention == slot
- slot is filled
- Head of query list is the slot
- N. of Unanswered System proposals
- Available User proposal
- Unavailable User proposal prior to a System utterance
- Unavailable User proposal after a System utterance
- Unavailable User assertion prior to a System utterance
- Unavailable User assertion after a System utterance
- N. of values that have not been proposed or gaveInfo on

## VI. TAU AND NSF DEMONSTRATION

We have presented the Tau architecture that separates Turn, Attention, and Utterance decisions, and NSF, a rich negotiation domain that models certain psycholinguistic phenomena. We have also sketched the Tau implementation in the NSF domain. Towards showing the advantage of flexible dialogue managers and the utility of the NSF domain in their development, we demonstrate both contributions by examining learnt policy differences as the dialogue environment changes.

We set the number of available values to be 3, and the total number of values to be 5. We then varied the size of the Goal Stack to be 3, 5, or 7. A shorter Goal Stack means the achievable combination has a greater probability of occurring earlier than with a longer one. We train the System for 100k Epochs (10m dialogues) and then test it with 50k dialogues. Epoch's are made up of a 100 dialogues and the policy is updated after every one. There is no exploration during test dialogues so this effectively tests the policy. We use the Q-Learning algorithm with an exploration rate of 20% (epsilon = 0.2). Q-learning, a dynamic programming algorithm, updates  $Q(s, a)$  based on the immediate reward and the best action of the next state which is not necessarily the one taken during the training dialogue.

### A. Results and Discussion

We report the percentage of bids used by the System over the test runs. In terms of turn-taking functionality, *silence* and *lowest* is indicative of trying to lose the turn where *highest* and *high* is indicative of *attempting* to take the turn. The other two turn-bids, *low* and *mid*, are indicative of trying to filter the turn, as described in our previous work [8]. This filtering enables User's who have a strong beliefs concerning their utterance to get the turn, and User's who have weak beliefs not to.

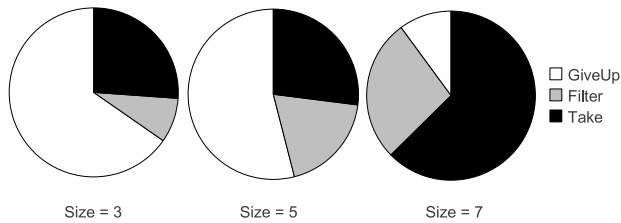


Fig. 2. Turn-taking move percentage as Goal Stack size increases

Figure 2 provides the results, binning the turn-bids in the functional groups for clarity. We see that as the Goal Stack gets longer the System increasingly tries to take the turn, and use filtering bids. By the time the Stack size is at 7, the System is trying to get the turn nearly 2/3 of the time.

Table II gives an example dialogue with two slots for each of the Stack sizes. These dialogues illustrate that the utterance selection differs as well. When the goal stack is small, the System seems to take a backseat roll and only speaks when the User gives an unavailable value. However, as the goal stack gets longer the System becomes more proactive and begins to attempt to give more information to the User. This is because the User is more likely to have unachievable combinations early in the dialogue. This proactive behavior is most certainly a function of the number of available values. If this number was increased than the size threshold before the System becomes proactive would also increase.

The turn-taking behavior with the smaller Goal Stack, where the System rarely attempts to actually take the turn when the

TABLE II  
DIFFERENT DIALOGUES AS THE GOAL STACK SIZE VARIES. (ONLY TWO SLOTS ARE SHOWN)

Line	Speaker	Utterance	System Bid
GS = 3			
1	User:	assert burger bacon	Give
2	User:	propose side salad	Give
3	System:	reject side salad	Take
4	User:	cancel burger bacon	Give
5	User:	assert side fries	Give
6	User:	assert burger regular	Give
7	...		
GS = 5			
1	System:	giveInfo burger regular	Filter
2	User:	assert burger regular	Filter
3	System:	giveInfo side fries	Take
4	System:	giveInfo side soup	Filter
5	System:	giveInfoLast side bread	Filter
6	User:	assert side fries	Give
7	...		
GS = 7			
1	System:	giveInfo side soup	Take
2	System:	giveInfo side fries	Take
3	System:	giveInfoLast side bread	Take
4	System:	giveInfo burger veggie	Take
5	User:	assert burger veggie	Take
6	User:	assert side fries	Give
7	...		

User could want it, is very similar to the Keep or Release turn-taking approach where the System will not speak until the User is done. However, this strategy would not be appropriate for longer Goal Stacks where providing timely information is key and the superiority of the System taking the turn is shown. The primary point is that flexible and dynamic turn-taking is important in rich mixed-initiative negotiation dialogues and the use of RL is far superior because of its ability to adjust to the environment. Furthermore, utterance selection differs as well and RL can jointly optimize each for the specific situation. In sum, this demonstration has shown not only that NSF is a rich and complex simulation to develop dialogue managers but that the joint optimization of all three decisions can lead to substantially different System behavior as the environment changes.

## VII. CONCLUSION

We have presented two contributions towards the development of mixed-initiative dialogue agents. We first described Tau, an IS architecture which has three separate actions for Turn, Attention and Utterance decisions and illustrated the use of RL in dialogue systems for learning these three actions jointly. However, training RL agents can be difficult and simulation is required during development before evaluating a dialogue management framework on real Users. Towards the goal of better simulations, in our second contribution we presented the NSF domain for RL dialogue management development. We then demonstrated both of these contributions and further highlighted the flexibility advantage learning dialogue management architectures have. We hope to continue the development of mixed-initiative dialogue systems using the

Tau dialogue management architecture, and this work provides a solid foundation for such pursuits.

### Acknowledgments

We gratefully acknowledge funding from the National Science Foundation under grant IIS-0713698.

### REFERENCES

- [1] A. Gravano and J. Hirschberg, "Turn-yielding cues in task-oriented dialogue," in *Proceedings of the SIGDIAL 2009 Conference: The 10th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. Association for Computational Linguistics, 2009, pp. 253–261.
- [2] R. Levitan, A. Gravano, and J. Hirschberg, "Entrainment in Speech Preceding Backchannels," in *Proc. of ACL 2011*, 2011.
- [3] D. Bohus and E. Horvitz, "Multiparty Turn Taking in Situated Dialog: Study, Lessons, and Directions," in *Proc. of SIGdial 2011*, 2011.
- [4] T. Baumann, M. Atterer, and D. Schlangen, "Assessing and improving the performance of speech recognition for incremental systems," in *Proc. NAACL: HLT*. Association for Computational Linguistics, 2009, pp. 380–388.
- [5] H. Sacks, E. Schegloff, and G. Jefferson, "A simplest systematics for the organization of turn-taking for conversation," *Language*, vol. 50, no. 4, pp. 696–735, 1974.
- [6] A. Raux and M. Eskenazi, "A finite-state turn-taking model for spoken dialog systems," in *Proceedings of HLT/NAACL*. Association for Computational Linguistics, 2009, pp. 629–637.
- [7] G. R. Jonsdottir, K. R. Thorisson, and E. Nivel, "Learning smooth, human-like turntaking in realtime dialogue," in *IVA '08: Proceedings of the 8th international conference on Intelligent Virtual Agents*. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 162–175.
- [8] E. Selfridge and P. Heeman, "Importance-Driven Turn-Bidding for spoken dialogue systems," in *Proc. of ACL 2010*. Association for Computational Linguistics, 2010, pp. 177–185.
- [9] K. Scheffler and S. Young, "Automatic learning of dialogue strategy using dialogue simulation and reinforcement learning," in *Proceedings of the second international conference on Human Language Technology Research*, 2002, pp. 12–19.
- [10] O. Lemon, K. Georgila, J. Henderson, and M. Stuttle, "An ISU dialogue system exhibiting reinforcement learning of dialogue policies: generic slot-filling in the TALK in-car system," in *Proceedings of the Eleventh Conference of the European Chapter of the Association for Computational Linguistics: Posters & Demonstrations*. Association for Computational Linguistics, 2006, pp. 119–122.
- [11] K. Georgila, M. Wolters, and J. Moore, "Learning dialogue strategies from older and younger simulated users," in *Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. Association for Computational Linguistics, 2010, pp. 103–106.
- [12] S. Larsson and D. Traum, "Information state and dialogue management in the trindi dialogue move engine toolkit," *Natural Language Engineering*, vol. 6, pp. 323–340, 2000.
- [13] P. Heeman, "Combining reinforcement learning with information-state update rules," in *Proceedings of the Annual Conference of the North American Association for Computational Linguistics*, Rochester, NY, 2007, pp. 268–275.
- [14] E. Levin, R. Pieraccini, and W. Eckert, "A stochastic model of human-machine interaction for learning dialog strategies," *IEEE Transactions on Speech and Audio Processing*, vol. 8, no. 1, pp. 11 – 23, 2000.
- [15] R. Sutton and A. Barto, *Reinforcement Learning*. MIT Press, 1998.
- [16] L. Busoniu, R. Babuska, and B. De Schutter, "A comprehensive survey of multiagent reinforcement learning," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 38, no. 2, pp. 156–172, 2008.
- [17] P. Heeman, "Representing the Reinforcement Learning state in a negotiation dialogue," in *Automatic Speech Recognition & Understanding, 2009. ASRU 2009. IEEE Workshop on*. IEEE, 2009, pp. 450–455.
- [18] C. Sidner, "An artificial discourse language for collaborative negotiation," in *Proceedings of the National Conference on Artificial Intelligence*. JOHN WILEY & SONS LTD, 1994, pp. 814–814.
- [19] M. Walker, "Inferring acceptance and rejection in dialog by default rules of inference," *Language and Speech*, vol. 39, no. 2-3, p. 265, 1996.
- [20] J. Chu-Carroll and S. Carberry, "Response generation in collaborative negotiation," in *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*. Morristown, NJ, USA: Association for Computational Linguistics, 1995, pp. 136–143.