Flexible dialogue management and cost-models

Ian Lewin
University of Cambridge Computer Laboratory
email: ian.lewin@cl.cam.ac.uk

Abstract

Flexibility in dialogue management requires not just the ability to understand and respond to a greater range of user utterance types (or *moves*), but also the ability to generate them and to do so strategically in accordance with some notion of costs and benefits. We explore this issue in the context of the Information State Update model of dialogue. We add costs and preferences to a simple instantiation of the model and explore the added flexibility this brings and also link the inclusion of costs to other developments of the model. We compare this work to the work in reinforcement learning which also includes a notion of cost and reward.

1 Introduction

The Information State Update (ISU) approach to dialogue modelling is a highly abstract characterization of dialogue semantics. Contributions to a dialogue are treated like programs in dynamic logic: they both *update* a dialogue state and are *interpretable* in the light of a previous state.

Agents have two main roles in this abstract picture: to use state in interpreting contributions; and to make state by generating contributions. A great deal of research has concentrated on the former question. What must a state look like if I am to be able to interpret *this* sort of conversational offering? And what will it look like once I have both interpreted and incorporated it? I want to ask: what must a state look like, if I am to choose to make *this* sort of conversational offering? And what will it look like once I have made it? My focus will be on the role of the dialogue state in spoken dialogue systems, partly because this is a useful (and largely externally imposed) constraint on the extent of the material to consider, partly because the results may be practically useful.

The ISU approach to dialogue modelling easily accommodates dialogue models that are encodable directly in a network formalism with atomic states and transitions between them. Choices over dialogue moves can be encoded in a nondeterministic network. The frame based (or slot-and-filler) architecture, for example as instantiated in the form interpretation algorithm in the commercially employed VoiceXML dialogue specification (Oshry et al., 2006) is also easily implementable. An example frame for the travel planning scenario is shown in figure 1. The different instantiations of the frame form the states of the system. Dialogue policy is implemented by associating a question with each attribute in the frame and by ordering the attributes. The next question to be put is the first attribute for which no value is currently known. The whole frame may also be associated with a question, which is put when a frame first comes into focus. The frame stays in focus until all attributes have received values. The VoiceXML question selection algorithm is: if the frame is already in focus, ask the first question (from top to bottom) whose value is unknown else ask the question associated with the whole frame.

The general slot-and-filler approach has continued to underpin a large amount of research in dialogue systems including theoretical work that is implemented in demonstration systems and in systems

that attempt to learn dialogue strategy. In the next section, we review this work and highlight the need for informing machine learning approaches with insights from theory as well as the need for the theoretical approaches to include explicit cost models. Following this, we add a simple cost and preference model to an Information State Update based dialogue manager and explore some of its implications. With a focus on practical systems, we consider the dialogue management of questions that are not directly answered, comparing the treatment with that of models which add features such as "questions under discussion". We conclude that the addition of a cost model is vital not only for future theoretical work but also as a basis for informing future targets for machine learning approaches.

2 Previous Work

Dialogue strategy development in the Information State Update approach has followed two main trends. Theoretical work has concentrated mostly on extending the notion of dialogue state in order to permit analysis of a greater range of dialogues and a deeper analysis of phenomena such as grounding. Secondly, there has been a strand directed towards learning dialogue strategy automatically from real or simulated dialogues. For computational reasons, this work has tended to use a much narrower conception of state. Indeed, the focus of attention has almost entirely been devoted to learning whether and how to confirm user utterances and whether the next question should be more open-ended or not ("mixed initiative"). Although the notion of state in this work tends to be more limited, the strategy is clearly linked to notions of costs and progress towards dialogue goals.

Chu-Carroll and Nickerson (2000) and Litman (2002) describe experiments showing that online adaptation in dialogue strategy is beneficial to users. Litman and Pan's TOOT system monitors predicted speech recognition error rates and can change strategy twice during a dialogue. The system begins by not confirming user utterances at all. If problems are detected, it can start implicitly confirming utterances ("I heard you say Sunday. What time would you like to leave?"); and if problems continue, it can move to explicit confirmation ('On which day of the week do you want to leave", "Sunday", "Do you want to leave on Sunday?"). Simultaneously, the open-endedness of the prompts and the range of acceptable responses is degraded. For example, at the middle stage (entitled *mixed initiative*, somewhat bizarrely) the system will not let the user ignore its question but insists on an answer. Chu-Carroll and Nickerson's MIMIC system also monitors more general interpretation difficulties than just speech recognition problems. A binary global switch can be set causing the system to offer less open-ended prompts. The switch may also be reset if sufficient cue evidence of success can be found. One cue is whether the user subsequently adds unsolicited information to an answer.

Reinforcement learning techniques have also been widely explored as a means of data-driven development of dialogue strategy. In addition to a corpus of dialogues encoding dialogue states and transitions between them, a reward function is required which enables the learning function to generate a strategy that generally leads to higher rewards.

For example, Scheffler and Young (2002) used a corpus of dialogues generated with a user simulation to learn how to confirm what was just said (explicitly, implicitly by repeating whilst querying the next slot or not at all) and whether to offer open-ended questions or not. In addition to slot information, the state contained confidence scores for the latest recognition result. Scheffler and Young argue that it is appropriate to restrict the learning to suitably *local* decision making (such as confirmation of last utterance) on the grounds that the user simulation provides the data for making these choices but not on the higher strategic questions such as which slot to ask for next. In contrast, Henderson (2005) explores the learning of strategies for the entire dialogue and also uses a much richer notion of dialogue state including possibly its entire history. The learned strategies are reported to perform well when tested against user simulations and against real users (Lemon et al., 2006). Qualitative analysis of the dialogues (Frampton and Lemon, 2006) suggests that the improved performance of the learned strategies is actually entirely attributable to improved (local) repair strategies when the last exchange failed to add a value for a slot. Rather than simply repeating a failed question the improved strategies would either

give help or switch focus to a different slot. Giving help is an evidently sensible strategy. The switch of focus to another slot is interesting however not least because the reward function only rewarded dialogues where all slots are filled and the learner did learn only to query the database when all slots had been filled. Consequently, switching focus could only postpone acquisition of a value for a particular slot. The system apparently learned that exact question repeats tend to be unsuccessful. In general, question repeats may be unsuccessful if answer pronunciation remains identical or indeed if the answer becomes hyper-articulated. Possibly these properties were reproduced in the n-gram user simulations used for testing which were trained from Communicator data. In this case, asking the same question again later but not actually immediately might be more likely to result in a recognizable response. However there is clearly a danger that the learned strategies are just responses to somewhat unnatural artifacts of the user simulation. Whether such a strategy could actually be viable for real human users is an open question. One obvious alternative is simply to ask for the same information again but in a rather different way. However, the set of possible actions in the learning experiment did not include this particular move.

Pietquin and Renals (2002) earlier trained a slot-filling system in which not all slots were required for a database query and generated a strategy which preferentially asked for slots whose values were more likely to be recognizable - a seemingly simple and effective policy which later researchers do not seem to have pursued.

Re-raising questions differently has also been proposed from the more theoretical strand of work using the Information State Update approach. Cooper and Larsson (2005) maintain a Qud-like stack of questions in the Information State which have been raised but not yet resolved. One possible use of this is so that later question repeats could be reformulated. Another is to allow interpretations of material which are not interpretable "alone" but require the earlier question as context. If the question remains suitably salient even though it is not the latest utterance then the material can be resolved. There is also the possibility of an accommodation mechanism in which material that requires a question to be salient

$$\begin{bmatrix} \text{FROM} \\ \text{DEPARTURE-TIME} \\ \text{TO} \\ \text{MODE OF TRAVEL} \end{bmatrix}$$

Figure 1: Travel Frame

1. [S] When do you want to leave?
2. [U] *I want to be in Gothenburg at 8.*

Figure 2: Unsolicited information

in order to be interpretable at all actually causes that question to become salient. Although demonstration systems have been built that are capable of illustrating these principles, it is far from clear that they are sufficiently robust to permit real dialogues to proceed smoothly and transparently to successful conclusions. Furthermore, discussions of these systems tend not to be explicit about how agents cost and select their moves.

3 Non-answers to questions

The key merit in the VoiceXML form interpretation algorithm is simply that it easily permits appropriate future dialogue behaviour upon encountering unsolicited relevant information, as in figure 2.

If "I want to be in Gothenburg at 8" can be understood as a possible answer to "How can I help you?" or even as an initiating exchange utterance on its own, then it ought also to be processable as a response to "When do you want to leave?". The user response is not an answer, not a direct one at least, but it can be used to fill in the value of a slot nonetheless. Then, the next question to be asked can be calculated as before on the basis of what else the system needs to know. The default VoiceXML strategy will end up repeating the very same question. The focus switching strategy (see above) will also repeat the very same question again only perhaps not just yet.

What is required for an agent to select more intelligently amongst the options it actually has available? In what circumstances would an agent move on to a different question. An agent rea-

sonably requests the same information (possibly via a re-phrasing) if the answer is either *essential* for progress; or if, more mundanely, it just remains the best bet for making progress. One advantage in concentrating on a typical task for a spoken dialogue system is that reasonable measures are easier to come by. If the immediate goal is to make a database query then an "essential" piece of knowledge is one without which no query can be made. A dialogue manager which insists on asking a particular question even though the underlying query system does not require it is clearly deficient. If the question is not essential, then the likelihood of making progress through a repeat needs to be *weighed*. Theoretical models stand in need of an explicit cost model and evaluation procedure in order to meet this demand.

Given the simplest slot-and-filler information state (Figure 1), one very simple addition to enable more intelligent selection is to just define a preference ordering over all the subsets of possible attributes e.g. $Sel : Pow(A) \rightarrow Z^{+}$. The selection algorithm then becomes: choose the attribute whose addition to those already supplied with values maximizes the value of Sel. Such a function is very easily implemented. Depending on the particular Sel function defined, a dialogue manager can now

1. repeat the original question
2. ask a different one
3. execute the database query straight away

If the system knows X and requests M, but receives N, then (X+N+M) might be considerably lower than (X+N+O) for some other O, so M should not be repeated but replaced by O. Indeed, X+N might be better than any extension of it, in which case no further questions should be asked: the intended database query can be made now. If we further add in a simple cost function which progressively penalizes repetitions of questions, then we essentially have the system of (Lewin, 2001). Another addition to the cost penalty might be the likelihood of receiving a recognizable answer, in the style of (Pietquin and Renals, 2002).

The cost and evaluation model can therefore be used to build in certain simple dialogue strategic principles. For example, the simple cost on repetitions allows repeats if the information sought is sufficiently important. Equally, there may be many sets of attributes which cannot be extended to a more preferred set. That is, there may be many different ways of achieving the goal of making *some* database query. Not all travel queries will require finding out the departure-time. Other general constraints might be imposed on the preference ordering. In general, more query constraints are better than fewer, because the set of satisfiers will likely be smaller and it will be easier to discuss and evaluate a small set of alternatives later. On the other hand, a query that is likely to return no answers (because the query is overly specified) might receive a very low value indeed.

Is this not precisely the sort of information that might be built into a reward function for a reinforcement learner? The answer of course is "yes" and "no". In principle, everything can be built in. In practice, not everything can be. Furthermore, what is built in in practice needs to be guided properly by theory. We have seen one instance of this already in which the learner learns to repeat a question later given that it prefers not to repeat it immediately but does actually need the answer to progress. Another highly plausible scenario is this: the travel options for Gothenburg can change over time and thus the best way to make progress in dialogues about those travel options can change over time too. As more travel options become available, it might become sensible to ask about the travel-mode earlier in the dialogue in order to reduce the size of the response from the database query. It is certainly feasible that a learning algorithm could learn automatically the changing relative costs of different travel queries; and these values could then function as an *input* to a dialogue management algorithm such as that sketched above. In general, the point is this: the development of theory which has practical ambitions needs to incorporate a cost model; furthermore, these developments can usefully *inform* further efforts at deploying machine learning. The point resembles that of Scheffler and Young: one need not attempt to model all possible dialogues but restrict the learning to parameters that one can effectively obtain data for.

The general analysis is not restricted to this partic-

ular scenario in which progress is evaluated by anticipated results of a database query. One might, for example, be able to execute a query after every new piece of information from the user and thereby have access to the actual travel possibilities given what the user has said so far. In this case, the next question might be calculated using Information Theoretical considerations: which question is the best next one to ask to identify the right travel option out of this set of possibilities? Again, values for such a parameter might be trainable from data without having to retrain the entire dialogue manager.

3.1 Stacks and Quds

Theoretical work in the Information State Update approach has often placed emphasis on storing questions (or issues) in stack like structures.

When the dialogue manager interprets the user's unsolicited offering and calculates his next move, need it note that a question was asked but not answered? Do we pop the question off a stack? The system knows of course which goals remain unsatisfied so the issue here is not whether to put the question again or not; just whether to note that there is an outstanding question that has not been answered. The distinction between public shared parts of dialogue state and private personal ones becomes important here. One intuition in the theoretical work is that the putting of a question is a public act that alters a shared linguistic environment; and this is independent of any mental states that might have led to its putting. Unfortunately the public or private status of an "outstanding" question is far less clear. Even in our simple extract (figure 2), the response might be a simple rejection of the question, in which case it is unclear if the original question remains outstanding or not. Alternatively, perhaps the user does intend to return to the original question at some point in the future and has placed the issue on a mental stack. Cooper and Larsson (2005) note how subtle a decision on this question might be.

I want to make two points about this issue. The first is that whatever our *best* interpretation of the user's offering, our next action should not simply be determined by that interpretation plus our own plans and goals. A rational cost calculation must include not only the risk of our best interpretation being incorrect but the possible cost of the next possible actions. How much will next actions differ if we think the user was rejecting our question or just postponing it. At best, we will only have a probability distribution over interpretations since the actual state of the user is of course unobservable. (Williams and Young, 2005) have recently been exploring the use of partially observable Markov decision processes in the settings of spoken dialogue systems.

The second, and strongly related, point is to emphasize that whether the dialogue "needs" to pop a stack or not is partly a matter of what strategic calculations about dialogue progress the dialogue manager *can* make. The dialogue manager, when interpreting the user's unsolicited offering, is of course also just about to make another contribution to the dialogue itself. If it is really only capable of asking questions at this point, then whatever the current state, the new most salient outstanding question will be the one it now chooses to make. Could the original question be required as context for a subsequent elliptical utterance? This is perhaps not impossible although it is unlikely given that the original question was not answered and has now been superseded by a new one. The original question is otiose as context if the question is just repeated of course.

The most important reason to record the original question on a stack is simply if the moves available to the dialogue manager include one that explicitly hands the initiative to the user. It is a common enough human strategy, if a non-answering response is made to a question, only to acknowledge the response. One might either signal, perhaps through intonation, that more input is expected or just wait and see what turns up next. This is not just a characteristic of chit-chat conversations, in which it doesn't particularly matter what happens next. It can be a calculated move in a strictly goal-driven interchange. In a chess game, one player might believe he understands his opponent's plan of attack, but play an inconsequential move to allow his partner the opportunity of making a more revealing move. Of course, such a move is a luxury item in the current state of spoken dialogue systems! Theoretically, however, the point is that "only acknowledging" is itself a particular dialogue move with consequences that need to be weighed in a model of move selection. Furthermore these possibilities are intimately tied to the addition of stack-like structures to the in-

1. [S] Do you want the train or the plane?
2. [U] *What time do they arrive?*

Figure 3: Dependent questions

1. [S] Do you want the train or the plane?
2. [U] *What time do they arrive?*
3. [S] The train arrives at 3 and the plane at 4
4. [U] *The train please*

Figure 4: Long distance short answers

formation state. If a dialogue manager is to be *capable* of just acknowledging a user assertion, then it ought also to be sensitive to the cost of doing that versus something else. Clearly, practical dialogue managers with this sort of capability have yet to be built. Theoretical accounts that include acknowledgments often appear to include them simply as part of a "protocol of interaction".

4 Questions in answer to questions

Stacks and Quds are popular also for analysis of dialogues that include nested questioning sequences (figure 4). Classically, these cases are ones where the answer to the first question somehow depends on the answer to the second. If it is analysed as being dependent, then the first question may be put on a stack; or perhaps a nested conversational game is begun, or possibly an outstanding obligation to answer it is recorded (Ginzburg, 1996; Lewin, 2000; Matheson et al., 2000).

Shifting the focus of our attention onto next move selection again urges rather the importance of weighing the possible next moves. What should a dialogue manager do when it receives a question in a response to question? Clearly, if the goal motivating the original question is still of primary importance, then it might be best simply to re-ask it (possibly with a re-phrasing). Alternatively, perhaps the question can be re-asked whilst also answering the new question. Finally, perhaps it is best simply to move onto another topic altogether. As we saw earlier with acknowledgments, it would be a mistake to suppose that the new question must simply be answered because of a protocol of interaction. In fact, the tactic of *only* answering the new question is just one further specific move with its own specific advantages and disadvantages that need to be weighed.

Does the dialogue manager need to record the original unanswered question? Again, the first point to make is that the original goal is still outstanding so the issue is one of noting an unanswered question, rather than whether to re-ask it. In the current case, it is clearer that the question might function as a context for a possible later ellipsis resolution; and this is arguably what happens in figure 4. Nevertheless, the matter is again intimately tied to the dialogue manager's choice of move, namely *only answering* the interjected question. The question of what to include in the dialogue state is not independent of what choices of action the dialogue manager can make. Practically speaking of course, it is worth remembering that correct ellipsis resolution is in any case not a simple operation. It is also increasingly hard the longer the distance grows between context and ellipsis and that, in many situations, there may often be other ways to achieve the right interpretation. Perhaps in most travel scenarios, "the train" only ever means "I'd like to travel by train".

Ginzburg, in his careful theoretical analysis (Ginzburg, 2007), also notes that dialogue stack structures do not arise as the results of seemingly arbitrary rules of dialogue syntax but from underlying dependencies amongst issues. That is, the answer to "train or plane" may depend upon arrival times. He goes further and states that "Since B can be assumed to attempt to provide information about $q1$ were she able to do so, the fact that she has not done so must mean that B cannot do so without (attempting to) resolve $q2$". It seems to me that this is just not true. Certainly in educational contexts, a teacher may respond to a question with another because this is just the best way to further the underlying educational objective even though a direct answer could have been given. This is not uncooperative behaviour. If a question is not answered, it does not follow that it could not have been answered, merely that the agent's choice of his next best move to further his own interests was not to answer it.

5 Cooperation and cost

A rather rich example of progress in a non-question answering dialogue is illustrated in Figure 5, a genuine extract from a travel agency dialogue conducted

1. [S] Välkommen till resebokningstjänsten. Hur jan jag hjälpa dig?
   [Welcome to the travel bureau, How can I help you?]
2. [U] Jag vill ha en resa till Göteborg från Stockholm.
   [I want to travel from Stockholm to Gothenburg.]
3. [S] Vilket datum vill du resa?
   [What day will you travel on?]
4. [U] Jag vill vara framme i Göteborg den elfte i åttonde.
   [I want to be in Gothenburg on August 11th.]
5. [S] Vill du resa med tåg eller flyg?
   [Will you travel by plane or train?]
6. [U] Vilken tid kommer dom fram?.
   [What time do they get there?]
7. [S] Vilken tid vill du resa ungefär?
   [What time will you leave?]
8. [U] Jag vill vara framme i Göteborg före klockan åtta.
   [I want to be in Gothenburg for eight o'clock]
9. [S] Det finns ett flyg som avgår klockan sju noll noll och ankommer klockan åtta noll noll. Det är det tidigaste flyget.
   [There is a flight that leaves at seven zero zero and arrives at eight zero zero. That is the earliest flight.]

Figure 5: Progress without Answers

between a human and a human (or "wizard") simulating a computer system. The wizard used a script to determine the next system utterance although it did not necessarily cover all conceivable circumstances.

The dialogue complexity increases rapidly after utterance 4. It is an interesting exercise to try to discern which questions remain open for discussion at which points and which cues humans might use to help decide. Could the behaviour be reproduced by a wholly automatic computer system in conversation with a human? What structures would it need to do it? How much inference would it need to employ? The answer is simple: it only requires the simple resources we have already sketched. Let us suppose the wizard is actually issuing a database query after each user utterance and choosing what to say next based partly on the size of the results of that query (the "solution set"). Starting at 5, the wizard determines that an answer to the question of travel mode will most likely reduce the size of the solution set the most. The user does not answer this question. Unfortunately, his offering, 6, also does not reduce the solution set size at all; although had he asked "How much do they cost?", the story might be different as there may be many fewer costs than travel options. The number of arrival times in the solution set may similarly be too high. What should the wizard do next? Repeating the travel mode question is a possibility; and is still presumably optimal with respect to solution set size; but there is a penalty for repeats. So, in this case, the next best question, 7, is decided upon. The user now offers 8. Response 8 also does not answer the previous question. But 8 does in fact reduce the size of the solution set sufficiently and this phase of the dialogue can successfully close through 9.

It is noteworthy that none of the three questions, 5,6 and 7 was actually answered. Yet the dialogue has succeeded. Furthermore, the suggested flight in 9 is not, I think, even an answer to questions 5, 6, and 7. Indeed, even if the user were to follow up with "I'll take it", it is far from clear that that is a response to 5, 6, and 7, as well as 8.

Why could motivate a user to follow up question 5 with his own question but then question 7 with a non-answering statement? It appears that the user's motivating goal was probably all along the desire to be in Gothenburg by eight o'clock on August 11th. He was happy to play along with the system's line of questioning so long as progress was also being made towards this goal. Question 5 put this in jeopardy. An answer to this question might inadvertently rule out the possibility of being in Gothenburg at the desired time. By asking for arrival times, the user planned to pick one before eight o'clock. This tactic failed. However, the system's next question also could not further his objective and so he decided not to play along with the system's strategy anymore. He decided to override the system's question entirely and state the hard constraint he had in mind. That is one possible interpretation at any rate.

Does the dialogue instantiate *uncooperative* be-

haviour? Certainly questions were left unanswered; but each agent did at all times attempt to advance his own goals and, as the two sets of goals were indeed related, a mutually agreeable path forwards could be found. Co-operation is simply something that can exist at different levels of activity and at different times. One can answer a question in a way that does not advance the task; just as one can advance the task by not answering a given question. Cooperation is not a complete package that one just buys into or not in any given conversation. Besides, a robust system needs to be able to cope with a mildly uncooperative human it encounters.

6 Conclusion

Explicit cost models form an essential part of a complete account of dialogue management in the Information State Update model. The information required in order to make dialogue moves is just as important as that required to interpret them. Furthermore building in certain interpretative capabilities (such as stacked questions for ellipsis resolution) actually depends on the set of moves that a dialogue manager can make (such as "only acknowledging"). Finally, the addition of cost models and the design of strategy is not exhausted by the possibility of machine learning scenarios. Indeed, good theory can help direct the machine learning towards acquiring valuable parameters more effectively.

Acknowledgments

Thanks to Telia Research AB, Vitsandsgatan 9, S-123 86 Farsta, Sweden for permission to use their corpus of travel dialogues.

References

J. Chu-Carroll and J. Nickerson. 2000. Evaluating automatic dialogue strategy adaptation for a spoken dialogue system. In *Proc. 1st NAACL*, pages 202–209.

R. Cooper and S. Larsson. 2005. Accommodation and reaccommodation in dialogue. In R. Bäuerle, U. Reyle, and T.E. Zimmerman, editors, *Presuppositions and Discourse*. Elsevier.

Matthew Frampton and Oliver Lemon. 2006. Learning more effective dialogue strategies using limited dialogue move features. In *Proc. ACL '06*, pages 185–192.

J. Ginzburg. 1996. Interrogatives: Questions, facts, and dialogue. In Shalom Lappin, editor, *The Handbook of Contemporary Semantic Theory*, pages 385–422. Blackwell Publishers.

J. Ginzburg. 2007. *Semantics and Interaction in Dialogue*. Studies in Computational Linguistics. CSLI Publications.

J. Henderson, O. Lemon, and K. Georgila. 2005. Hybrid reinforcement/supervised learning for dialogue policies from communicator data. In *IJCAI workshop: Knowledge & Reasoning in Practical Dialogue Systems*.

Oliver Lemon, Kallirroi Georgila, and James Henderson. 2006. Evaluating effectiveness and portability of reinforcement learned dialogue strategies with real users. In *IEEE/ACL Spoken Language Technology*.

I. Lewin. 2000. A formal model of conversational game theory. In *4th Workshop on the Semantics & Pragmatics of Dialogue: Gotalog 2000*, pages 115–122.

I. Lewin. 2001. Limited enquiry negotiation dialogues. In *Eurospeech 2001: Proc. 7th European Conference on speech communication and technology*, pages 2333–2336.

Diane J. Litman and Shimei Pan. 2002. Designing & evaluating an adaptive spoken dialogue system. *User Modeling & User-Adapted Interaction*, 12(2-3):111–137.

C. Matheson, M. Poesio, and D. Traum. 2000. Modelling grounding and discourse obligations using update rules. In *Proceedings of NAACL 2000*.

Matt Oshry, RJ Auburn, Paolo Baggia, Michael Bodell, David Burke, Daniel C. Burnett, Emily Candell, Jerry Carter, Scott McGlashan, Alex Lee, Brad Porter, and Ken Rehor. 2006. Voice extensible markup language (voicexml) 2.1.

Olivier Pietquin and Steve Renals. 2002. Asr system modeling for automatic evaluation and optimization of dialogue systems. In *Proceedings of ICASSP*, volume I, pages 45–48, Orlando, (USA, FL).

Konrad Scheffler and Steve Young. 2002. Automatic learning of dialogue strategy using dialogue simulation and reinforcement learning. In *Proc. 2nd Int. Conf. on Human Language Technology Research*, pages 12–19. Morgan Kaufmann Publishers Inc.

J. D. Williams and S. Young. 2005. Scaling up pomdps for dialog management: The "summary pomdp" method. *Automatic Speech Recognition and Understanding, 2005 IEEE Workshop on*, pages 177–182.