Flexible Turn-Taking for Spoken Dialogue Systems

Antoine Raux

January 2006

Thesis Proposal

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Maxine Eskenazi, Chair
Alan W Black
Reid Simmons
Diane J. Litman, University of Pittsburgh

Copyright © 2006 Antoine Raux

Keywords: Spoken Dialogue Systems, Turn-Taking

Abstract

Most of the research on spoken dialogue systems so far has focused either on higher levels of dialogue or on speech understanding. In contrast, the low-level interactional aspects of conversation such as turn-taking have been essentially ignored, leading builders of practical systems to resort to simple pause detection-based methods to handle turn-taking. In a preliminary study based on the Let's Go bus information system, I found that such methods lead to interaction failures and potentially to complete dialogue breakdowns for a significant proportion of the dialogues with real-world users. In addition, a comparison of conversational rhythm in successful dialogues showed that even when speech recognition is not a major obstacle to communication, systems perform very differently and much less efficiently than human speakers. To address these issues, I propose an approach that relies on two innovations over current dialogue systems. First, it features a new event-driven system architecture that allows real-time processing of conversation, which I implemented in the Olympus/RavenClaw spoken dialogue framework. In addition to the dialogue manager and the traditional understanding and generation modules, a new module, the interaction manager, is in charge of dynamically monitoring and managing low-level interaction phenomena. The second component of the proposed approach is the turn-taking model used by the interaction manager. Inspired by mobile robotics and autonomous agent research, the model is composed of a set of sensors that provide information about the world, a set of actions that the system can take, and an action selection mechanism. Although I will explore different such mechanisms, Reinforcement Learning appears to be an appropriate framework for learning turn-taking behavior. The last expected contribution of this thesis is in the form of an evaluation framework for turn-taking in spoken dialogue systems. This important aspect of the proposed work will include a study of various local and global metrics of turn-taking and dialogue, along with the design and validation of composite metrics. All the theoretical findings and models proposed in this thesis will be grounded and validated in real world applications including Let's Go and other RavenClaw-based dialogue systems.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

After several decades of research and development effort in the realm of practical spoken dialogue systems, the technologies have matured enough to allow wide spread use of such systems. Still, the approaches that have permitted the creation of working systems have left many issues unsolved. This makes spoken conversation with artificial agents sometimes unsatisfactory, seldom natural. Perhaps the most prominent issue is the quality of automatic speech recognition (ASR), which often results in misunderstandings that, in turn, lead to dialogue breakdowns. There are several approaches to overcome this problem: improving recognition technology for dialogues [Lemon, 2004], limiting the interaction in some way [Farfán et al., 2003], or endowing systems with error handling capabilities to smoothly recover from misrecognitions [Edlund et al., 2004, Bohus and Rudnicky, 2005b]. While the first approach, improving ASR, seems the most obvious direction, improvements have been incremental and there is little hope for the accuracy of speech recognition to reach human understanding level anytime soon. The last two strategies provide a way to cope with imperfect ASR, but they both come with a cost: they make dialogues longer either by only letting the user provide small amounts of information at a time (as in strongly system-directed dialogues), or by generating confirmation prompts (as in systems with error handling strategies). This would not be an issue if, in addition to issues in spoken language understanding, current spoken dialogue systems did not also have poor turn-taking capabilities. Indeed, the cost of an additional turn for artificial conversational agents, in time spent and/or disruption of the flow of the conversation, is much higher than what happens in human-human conversation. As pointed out in recent publications [Porzel and Baudis, 2004, Ward et al., 2005], this weakness comes from the fact that low-level interaction has to a large extent been neglected by researchers in the field. Instead, they have concentrated on higher-level concerns such as natural language understanding and dialogue planning.

Indeed, even as more and more complex tasks have been addressed, from database access [Ferrieux and Sadek, 1994, Denecke and Waibel, 1997] to travel planning [Walker et al., 2002] to guidance through procedures [Bohus and Rudnicky, 2002] to mobile robot control [Lemon et al., 2001], low-level interaction processes, such as turn-taking, have stayed by and large unchanged. Most systems use a pipeline architecture where the user's speech gets sequentially split into utterances, recognized, parsed, fed to a dialogue manager which produces a response that gets changed into natural language and synthesized. In this framework, each component waits for the previous one to finish before starting its own processing. This approach has some software engineering advantages: it is simple to build and existing components (speech recognizer, parser, etc) can be used as-is and chained together. On the other hand, it introduces serious limitations to the system. First, at a theoretical level, it does not match how humans process information in conversations, as has been shown by psycholinguistic studies [Brennan, 2000]. The fact that humans process and react to speech *during* utterances is also evident from everyday conversational phenomena such as backchanneling (when a listener vocally acknowledges that she is listening and/or understanding what a speaker is saying without interrupting the speaker's turn) or cooperative endings (when a speaker speaks the end of an utterance started by another speaker). Second, at a more practical level, systems built on a pipeline architecture lend themselves to various interactional problems, such as inappropriate delays [Ward et al., 2005], spurious interruptions, or turn over-taking (when the user and the system get "out of sync" [Porzel and Baudis, 2004]). A dialogue system architecture that allows real-time processing and reaction is thus essential for better interaction. It is not, however, sufficient. In human-human conversation, low-level interactional behavior tend to follow specific patterns or rules, as has been shown by extensive research done by, among others, conversation analysts [Sacks et al., 1974]. If they are to interact naturally, or even just appropriately, artificial conversational agents must therefore possess models of such behavior. These models need to be evaluated, compared, and potentially optimized, on actual conversations. Therefore, evaluation metrics must be defined to measure the quality of interaction regardless of the content of the dialogue. This evaluation problem in itself is difficult and, to date, no widely accepted solution has been found.

The goal of this thesis is to improve the interaction ability of spoken dialogue systems. Specifically, it aims at 1) building an architecture for real-time interaction management, 2) designing and implementing models of conversational interaction, and 3) proposing a task-independent evaluation framework for such models.

The rest of this proposal is organized as follows. The next chapter shows the weaknesses of current dialogue systems through the results of a study of turn-taking in a spoken dialogue system for the general public. A survey of related work follows. Chapter 4 details a new (already implemented) architecture for spoken dialogue systems that gives systems the ability to perform real-time processing and reaction. Chapter 5 describes a (proposed) interaction model integrated in the new architecture. Finally, Chapter 6 addresses the issue of evaluating interaction quality by proposing experimental protocols and evaluation metrics.

Chapter 2

Turn-Taking Issues in Spoken Dialogue Systems: Preliminary Study

2.1 Introduction

In order to assess the quality of turn-taking in real-world state-of-the-art spoken dialogue systems, a corpus of data was collected using Let's Go Public, a telephone-based bus schedule information system available to the general public in the Pittsburgh area. First, turn-taking failures were identified, i.e. cases where the system made a decision such as taking or yielding the floor at an inapropriate time. The analysis of this data reveals a number of gross errors committed by a system used in a public context and gives an estimate of the impact of these errors on the whole dialogue. Second, to evaluate the turn-taking behavior of the system at a finer level, a subset of reasonably successful dialogues has been analyzed, excluding cases where understanding errors are the main cause of unnatural turn-taking. We compare the timing of turn-taking in these dialogues with similar naturally occurring dialogues with a human operator. The results of this study show that turn-taking failures are frequent and often lead to complete dialogue breakdowns. In addition, even for successful dialogues, the interaction with a system is significantly less efficient than with a human operator both on the system side and on the user side.

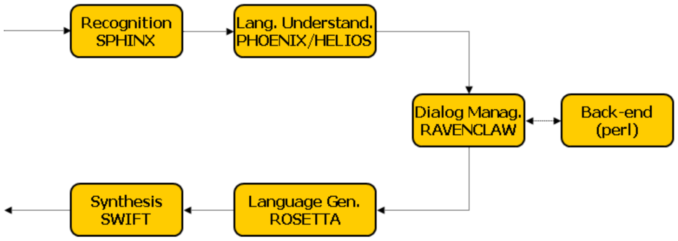


Figure 2.1: Architecture of the Let's Go Public spoken dialogue system.

2.2 Data Collection

2.2.1 Let's Go Public: A Dialogue System for Bus Schedule Information

To ground our work in a practical application, we built and deployed a dialogue system providing bus schedule information to the general public in Pittsburgh. The system uses the traditional pipe-line architecture represented in Figure 2.1. User utterances go through a speech recognizer and a parser, to the dialogue manager, which decides what response to give to the user. This response is translated from an internal semantic representation into English by a natural language generator and then into speech by a high quality limited-domain speech synthesizer. The system uses an adaptive energy-based end-pointer to detect user utterance boundaries. In addition, utterances that only contain noise (background voices, clicks, laughs, etc), as detected by the speech recognizer, are ignored even if their energy level is above the end-pointer's threshold. In a previous study using a mixed-initiative version of the system in a controlled experiment, the word error rate was 17% for native speakers of North American English and 43% for non-native speakers. In order to improve the robustness of the system before offering it to the general public, we modified the dialogue manager to have strongly system-driven interaction, making extensive use of explicit confirmation. Figure 2.2 shows part of a dialogue with this modified system. When designing the system, priority was given to task success rather than to naturalness. This approach is typical of a spoken dialogue system deployed for the general public. While, we do not claim that our present analysis will generalize to any spoken dialogue system, our goal is to show potential turn-taking issues with a state-of-the-art publicly deployed system.

	HC1	HC2	HH
Nb of dialogues	102	57	103
Total duration (in min)	436	144	127
Avg nb of turns/dialogue[1]	39.6	25.1	22.1
Word error rate	60%	28%	-
Task success rate	54%	93%	-

Table 2.1: Overview of the human-computer and human-human corpora.

2.2.2 Human-Computer Dialogue Corpora

The system described above takes calls from bus users daily, from 7pm to 7am on weekdays and from 6pm to 8am on weekends, when human operators are not at work. For this analysis, we used two subsets of calls received in early April 2005. A first set contains all the calls received on April 1st, 2nd and 3rd. There are 142 calls in total, of which 40 were excluded because they were too short (less than 5 user turns), or did not contain any speech directed to the system. This first set of 102 calls, hereafter referred to as the HC1 corpus, is a representative sample of calls to the system. A second set was assembled by collecting short dialogues (fewer than 15 user turns) in which the system provided information to the user. By applying this selection rule to the calls received between April 1st and 10th, we obtained 56 calls, which is about 12% of all the data collected during these 10 days. This set, hereafter called HC2, is representative of ”good” dialogues, where speech recognition performed reasonably well and no other major problem occurred.

Table 2.1 shows an overview of the two corpora. The success rate of the system on HC1 is 54%. This apparently low performance can be explained by the quality of the input speech (many calls are from people calling using cell phones in the street or with loud background noise), which is reflected in the overall word error rate of 60%. The sharp contrast between these figures and those from the controlled experiment reported in section 2.2.1 illustrates the differences between laboratory experiments and systems deployed ”in the wild”. The success rate in HC2, on the other hand, is 93% reflecting the effectiveness of the selection heuristic.

[1] These numbers include turns from both the system/operator and the user, not to be confused with the number of user turns used to select dialogues in HC1 and HC2

S: Welcome to the bus information system. This conversation will be recorded for technical research purposes. I am a voice-activated agent that can give you schedule information for some bus routes running in the East End, Downtown, and the airport. To get help at any time, just say Help or press zero. Which bus number or departure stop do you want information for?
U: 54C
S: The 54C . Did I get that right?
U: YES
S: Alright. Where are you leaving from?
U: OAKLAND
...
S: Leaving at 8 p.m. Is this correct?
U: YES
S: Right. Hold on. Let me check that for you. There is a 54C leaving FIFTH AVENUE AT BIGELOW at 8:19 p.m. It will arrive at EAST CARSON STREET AT TWENTIETH at 8:28 p.m. To get more information about buses related to this trip, you can say, when is the next bus, or, when is the previous bus. To ask about a different trip, you can say, start a new query. If you are finished, you can say goodbye.
U: WHEN IS THE PREVIOUS BUS
S: Okay. There is a 54C leaving FIFTH AVENUE AT BIGELOW at 8 p.m.
...

Figure 2.2: Excerpt from a dialogue with the system. (U: user turns, S: system turns)

2.2.3 Human-Human Dialogue Corpus

The third corpus (HH) contains 103 recordings of telephone calls to Customer Service human operators at the Port Authority of Allegheny County, which runs the buses. We selected them from a database of 3000 calls provided by the bus company, among calls that 1) dealt with scheduling information, 2) were less than 6 minutes long. These criteria were chosen to get dialogues that match, as closely as possible, dialogues with the system. In particular, we excluded calls unrelated to bus schedules, as well as very long calls which usually contain social interaction, small talk, and other out-of-task speech.

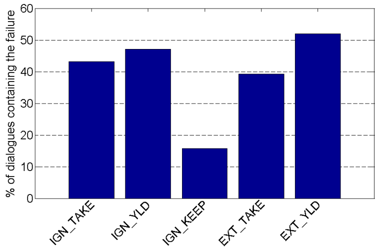


Figure 2.3: Frequency of Occurrence of Different Turn-Taking Failures.

2.3 Turn-Taking Failures in Human-Computer Dialogues

2.3.1 Frequency of Turn-Taking Failures

To better understand the nature and frequency of turn-taking failures in our system, we listened to each dialogue from the HC1 corpus and marked certain turn-taking failures as they occurred[2]. We distinguished five types of failures, based on three typical turn-taking actions: taking the floor (TAKE), keeping or holding the floor (KEEP), and yielding the floor (YLD). The labels are:

IGN_TAKE: the system ignores a user's request to take the floor during a system turn,

IGN_YLD: the system ignores the user yielding the floor, resulting in the system not responding after a user utterance,

IGN_KEEP: the system ignores the user keeping the floor, i.e. inadvertently barges in on the user,

EXT_TAKE: the system spuriously takes the floor, resulting in two successive system turns,

EXT_YLD: the system spuriously yields the floor, i.e. stops speaking in the middle of an utterance

Figure 2.3 shows the percentage of dialogues in which each of these failures occurred. The most frequent failure is EXT_YLD (52.0%), which is often due to background noise wrongly detected as speech, and misinterpreted as a barge-in attempt by the user. This can

[2] This annotation was based on recordings of the full conversation (both sides)

significantly harm the quality of the interaction, particularly when the user does not hear a question from the system, while the system waits for an answer. IGN_YLD occurred in 47.1% of the calls, again often as a consequence of background noise (e.g. street noise, music, voices), which prevent the end-pointing algorithm from reliably identifying pauses. This type of errors has a negative impact because it gives the user the impression that the system does not hear them, which results in changes in speaking style such as louder speech and hyper-articulation. IGN_TAKE occurs in 43.1% of the dialogues. Usually, this error is not as harmful as the previous ones because some users are aware that automated systems do not always allow barge-in, or they might simply repeat what they said. EXT_TAKE occurs in 39.2% of the dialogues, often in conjunction with EXT_YLD. Finally, IGN_KEEP occurs in 15.7% of the dialogues. This relatively low number compared to the other types can be explained by the fact that since the dialogues are system-directed, user utterances are short, with few pauses where the system can erroneously interrupt the user.

2.3.2 Dialogue Breakdowns due to Turn-Taking Failures

Among the 102 dialogues of the HC1 corpus, we identified the ones that failed at least partly because of turn-taking issues. A dialogue fits in this category if: 1) the user hung up before the system managed to provide them with the information they were looking for, and 2) at the time when the user hung up, there was an unsolved turn-taking failure. Of the whole HC1 corpus, 10 dialogues match these criteria, which amounts to 9.8% of the total number of dialogues and 21.7% of the failures. These task failures fall into two categories. For 3 of the calls, the conversation went normally up to a point where the system failed to detect the end of a user utterance, provoking a long pause (several seconds). Users apparently assumed the system was broken and hung up. For the 7 other calls, constant noise (background voices and in one case the caller breathing into the microphone) kept provoking IGN_YLD and EXT_TAKE failures. This has the double effect of giving the impression that the system does not hear the user and of potentially making the dialogue progress in the wrong direction without the user realizing it since system utterances are interrupted before their significant content can be said. The user doesn't hear the system's questions or request for confirmation and ultimately no interaction can take place because the system and the user are ”out-of-sync”.

While the small number of samples prevents us from deriving robust statistics on the frequency of different types of turn-taking breakdowns, there is clear evidence that they need to be addressed in order to maximize the dialogue quality and task success.

	Operator			Caller		
	Avg (s)	StdDev (s)	Nb Cases	Avg (s)	StdDev (s)	Nb Cases
Vocalizations	1.9	2.7	1423	1.5	1.5	1677
Simult. Speech	0.4	0.2	78	0.4	0.2	113
Pauses	1.6	2.9	330	0.9	1.5	641
Sw. Pauses	0.7	3.0	759	1.0	2.0	570

Table 2.2: Average State Duration (Avg) and Standard Deviation (StdDev) in the HH Corpus

	System			Caller		
	Avg (s)	StdDev (s)	Nb Cases	Avg (s)	StdDev (s)	Nb Cases
Vocalizations	1.4	1.6	2257	1.5	1.7	658
Simult. Speech	1.4	1.4	89	1.6	1.4	10
Pauses	1.6	2.1	1610	1.3	1.5	63
Sw. Pauses	1.5	1.8	549	1.6	3.2	582

Table 2.3: Average State Duration (Avg) and Standard Deviation (StdDev) in the HC2 Corpus

2.4 Comparison of Human-Human and Human-Computer Dialogue Rhythm

2.4.1 Data Analysis

The previous section focused on turn-taking failures by the system. However, even when no significant failure occurs, and even for successful dialogues, conversation with a system hardly has the ”feel” of natural conversation. Indeed, natural dialogues have an intrinsic rhythm implicitly negotiated by the two participants (see e.g. ten Bosch et al. [2004]), which human-computer dialogues lack. Without any rhythm, for the user, the interaction often amounts to a speaking-waiting-listening cycle that is all but natural and lacks the efficiency and flexibility of human-human conversation.

To evaluate the rhythmic differences between human-computer and human-human dialogues, we segmented and labeled the HC2 and HH corpora according to dyadic conversation states [Jaffe and Feldstein, 1970]. At each point in time, the dialogue is in one of 8 dyadic states, 4 for each speaker: vocalization (VOC: ”a continuous sound by the speaker who has the floor[3] ”), pause (PAU: ”a period of joint silence bounded by vocalizations of the speaker who has the floor”), switching pause (SWP: ”a period of joint silence bounded by vocalizations of different speakers”), and simultaneous speech (SSP: ”a sound by a speaker who does not have the floor during of a vocalization by the speaker who does”). In the following, we call ”a state” or ”a dyadic state” a stretch of time bearing a single label.

We labelled in two passes. First, we manually marked the regions where each participant was speaking. This gave us the floor possession information, along with the periods of simultaneous speech, which could not be identified automatically from a single-channel recording. Then, we semi-automatically detected the pauses within each speaker’s utterances. This second pass was performed using a Matlab script which looked for energy peaks within a 250ms window and marked the window as speech or pause based on an energy threshold. Because the recording conditions varied widely between dialogues, we manually adjusted the energy threshold for each dialogue so as to match as closely as possible our perception of pauses in the conversation. The final result of this phase is, for each dialogue, a sequence of dyadic states. Note that our state definitions are such that simultaneous speech is attributed to the speaker who has the floor at the time the overlap occurs, and that a switching pause is attributed to the speaker who had the floor before the pause.

2.4.2 Frequency of Occurrence of Dyadic States

As can be seen in Tables 2.2 and 2.3 the number of occurrences of the 8 dyadic states is not only different in absolute value (which can simply be explained by the different amounts of data in each corpus), but also in its distribution across states. Hence, while the number of occurrences of simultaneous speech and switching pauses for the operator/system are similar accross corpora, the number of pauses, and to a lesser extent of vocalizations, is significantly larger in system utterances. This reflects the fact that, by design, the system is more verbose than a human operator, often uttering several sentences at a time, and pausing between them. This is, for example, the case in the introduction and when giving results (see Figure 2.2). Another difference is that callers produce many fewer vocalizations and pauses when speaking to the system than to a human. This is the result of the constrained nature of the dialogue with the system, where users are only asked short an-

[3] Following Jaffe and Feldstein, we define ”floor” in the following way: ”The speaker who utters the first unilateral sound both initiates the conversation and gains possession of the floor. Having gained possession, a speaker maintains it until the first unilateral sound by another speaker, at which time the latter gains possession of the floor.”

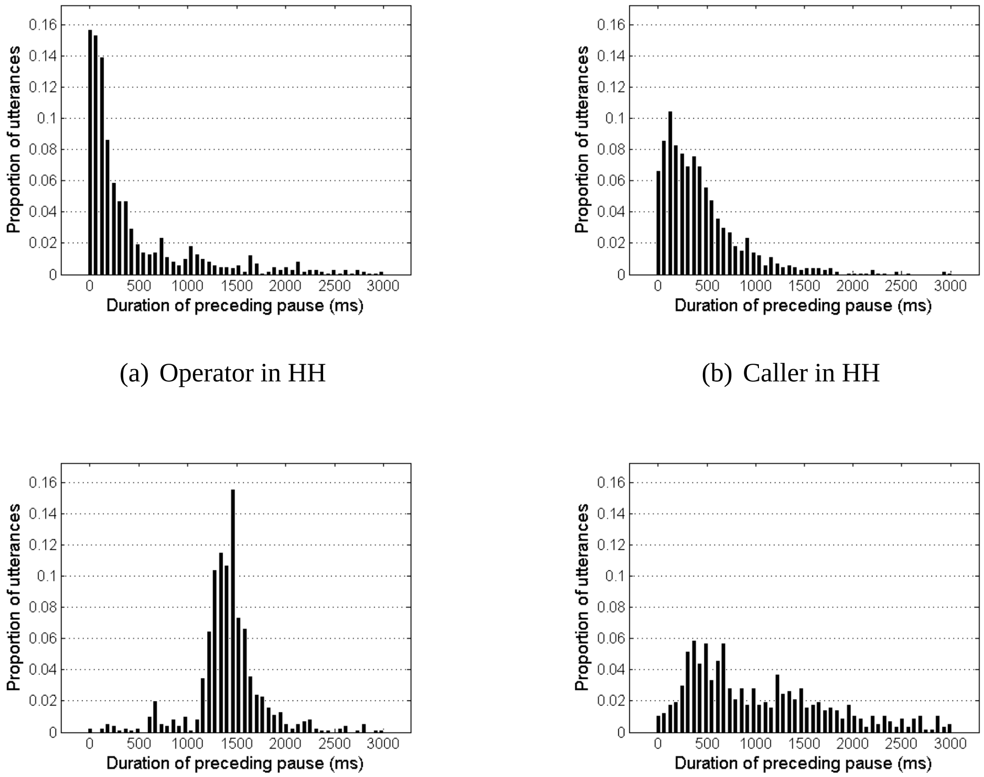
swer and yes/no questions. One can assume that a mixed-initiative system would yield longer user responses with more vocalizations and pauses.

2.4.3 Average State Durations

The differences in average duration between the human-human and human-computer corpora are statistically significant ($p < 0.001$ using a two-sample with unequal variance, two-tailed t-test) for all states except operator/system pauses and user vocalizations. In particular, periods of simultaneous speech are longer in the human-computer corpus, both for the system and the user. This reflects the fact that in task-oriented human-human conversations, overlapping speech is to a large extent limited to backchannels and very short overlaps at turn transitions. On the other hand, we have already observed that in many cases, the system fails (or does not attempt) to detect barge-in from the user, resulting in longer, unnatural, simultaneous speech. Second, pauses in user utterances are longer when speaking to the system rather than to a human. One explanation is that among the relatively rare cases where users do pause in their utterance, a significant number are due to the system failing to take a turn (IGN_YLD). The user then repeats their utterance or tries to reestablish the channel (e.g. by saying ”hello?”) after an unnaturally long pause. Finally, switching pauses are longer in the human-computer situation, both for the system (compared to the operator) and the user. Figure 2.4 shows the distribution of the duration of switching pauses for both participants to human-human and human-computer dialogues. First, it is clear that the system takes more time to respond than any other participant. Second, when talking to the system, human callers also take more time when they are dealing with a system than when it is a human operator. In addition, a closer look at this distribution of switching pauses preceding caller turns in HC2 shows that this ditribution appears to be bimodal. By looking at the data, we can explain this by the two most frequent types of answers that callers have to provide to the system: for answers to wh-questions (e.g. ”Where do you want to go to?”), the average preceding pause is 1341 ms, whereas for answers to confirmation questions (e.g. ”You want to go to the airport. Did I get that right?”), the average duration is 574 ms.

2.4.4 Interaction between Discourse Structure and Turn-Taking

To better understand the nature of the differences in switching pauses between the HH and HC2 corpora, we labeled system/operator utterances with dialogue moves, which were grouped into four main discourse functions: opening, initiation, response, and closing. Initiation and response moves follow the classification proposed by Carletta et al. [1997]



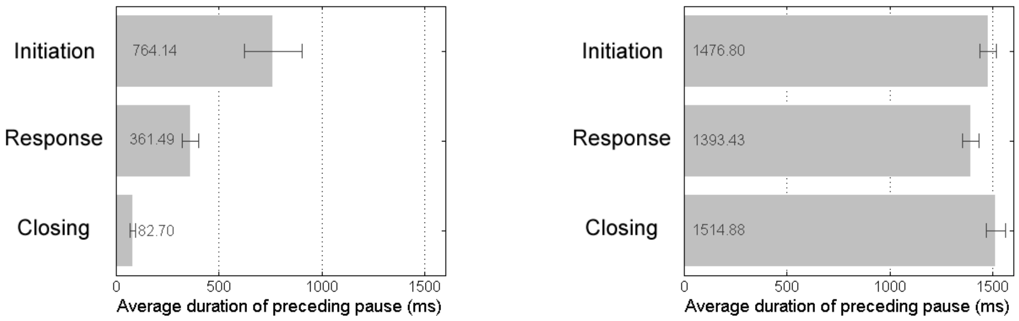
(a) Operator in HH

(b) Caller in HH

(c) System in HC2

(d) Caller in HC2

Figure 2.4: Histograms of the duration of the switching pauses preceding utterances by one of the participants in the HH and HC2 corpora



(a) Operator in HH

(b) System in HC2

Figure 2.5: Average duration (with error bars) of pauses preceding different types of dialogue moves

and already used to study the timing of turn-taking in Matthew Bull's thesis [Bull, 1997]. An initiation move is an utterance that introduces new expectations to the dialogue (e.g. asking a question, setting a goal). A response move is one that attempts to fulfill such an expectation (e.g. acknowledgment, response to a yes-no or wh- question). Additionally, we distinguish opening and closing moves (which correspond to utterances such as ”Hello” and ”Thank you”, ”You're welcome”, and ”Goodbye”) from other moves because they typically do not imply any transfer of information and correspond more to the ”protocol” people use when talking over the phone. In particular, because we are interested in switching pauses occurring before different types of moves, we will not address opening moves here. When an utterance contained several successive moves (e.g. an acknowledgment followed by a yes-no question), we used the move that appeared first since we assume that the pauses immediately preceding a turn are mostly affected by what comes first in the turn.

Figure 2.5 shows the average duration of pauses preceding different types of move for the operator in HH and the system in HC2. Besides the previously noted fact that the operator is always faster than the system, these graphs show that the response time of the operator depends greatly on the type of dialogue move, which is not the case for the system. The result on HH is similar to what Bull and Aylett [1998] found on the HCRC Map Task corpus, namely that inter-speaker intervals are longer between conversational games (i.e. before initiation moves) than within games (i.e. before response moves). In addition, we find that closings require the shortest response time and by far (almost a tenth of initiation moves). This is probably due to their fixed nature (thus not requiring much planning from the speaker). On the other hand, system response time in HC2 is by and large dictated by processing that is independent of the structure of the dialogue (e.g. endpointing, dialogue state computation, synthesis...). This results in almost constant reponse times.

2.5 Discussion

Our experimental results suggest that the current approach to turn-taking in spoken dialogue systems can lead to suboptimal interaction. First, many turn-taking failures occur in human-computer dialogues, and most of them would only appear in human-human conversation in the presence of a severely degraded channel (e.g. so noisy that one cannot always tell whether the other participant has finished speaking or not). Second, endpointing mechanisms based on pause detection alone result in significantly longer response times than human performance. Third, whereas human-human conversation presents a rhythmic pattern related to its meaning and discourse structure, our system (and, to the best of our knowledge, all other dialogue systems today) have mostly constant reaction time. The extent to which this fixed, unnatural rhythm hurts the dialogue (e.g. by annoying the user) remains an open question. In any case, this rigidity reflects the fact that turn-taking mechanisms in dialogue systems do not use information from the higher levels of conversation (structure and meaning).

The turn-taking behavior of human callers is also significantly affected by that of the system. There are two explanations for this. One is that humans adapt to the rhythm imposed by the system consciously or unconsciously, as has been observed by Darves and Oviatt [2002]. This, in itself, is not a problem as long as user adaptation to unnatural patterns does not hurt the quality of the interaction. A second reason for the difference in human response time could be that users need more time to process system utterances than human operator utterances. This could be due to the fact that system prompts are worded in an unexpected way. Also, some system questions, even if they are naturally worded, could come at unexpected times for the user, for example following misrecognitions (as in the following example dialogue: ”User: I want to go to the airport. System: Going to downtown. Did I get that right? User: ...”). The quality of speech synthesis, in particular its prosody, could also make comprehension and endpointing more difficult for users.

Chapter 3

Related Work

3.1 Turn-Taking Issues with Current Dialogue Systems

The vast majority of spoken dialogue systems use a pipeline architecture similar to the one represented in Figure 3.1. Even systems based on multi-agent architectures such as Galaxy-II [Seneff et al., 1998] or Open Agent Architecture [Cheyer and Martin, 2001] process information sequentially most of the time, passing the result of one agent to the next, with little or no parallel processing [Seneff et al., 1999, Stent et al., 1999, Rudnicky et al., 1999, Bos and Oka, 2002]. The Automatic Speech Recognition (ASR) module is in charge of endpointing (usually based on pause detection in the audio signal). Once it has detected the end of an utterance, it sends the recognition hypothesis to a Natural Language Understanding (NLU) module, which converts it into some semantic representation of the utterance. This is, in turn, used by the Dialogue Manager (DM) to update the dialogue state and decide the system's response. This response, produced by the DM in semantic form, is converted to natural language by the Natural Language Generation (NLG) module and synthesized and played back to the user by the Text-to-Speech (TTS) module. While differences exist between the various systems in the number and exact function of modules, each utterance basically ”traverses” the system in the way described above.

Several researchers have pointed out the short-comings of this architecture. For instance, Porzel and Baudis [2004] cite *turn overtaking* as a major obstacle to usability in state-of-the-art conversational dialogue systems. Turn overtaking happens when the user produces a second utterance right after a first one. This is due either to users correcting themselves or adding new information, or to the endpointer wrongly identifying a hesitation as the end of an utterance. The result is that the system considers the second utterance

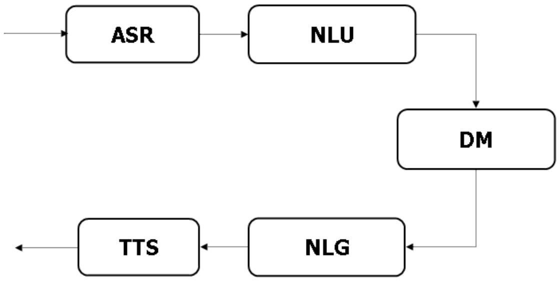


Figure 3.1: Standard Pipeline Architecture for Dialogue Systems.

as a response by the user to the system's response to the first utterance, although the user actually has not yet heard this system response. If no remediation mechanism exist (as in most of the cases), the dialogue then becomes ”out-of-sync”, i.e. the system always responds to the user's second to last utterance instead of the last one (because utterances are queued in the pipeline architecture).

In an investigation of usability issues and time losses in dialogues with a system compared to those with a human, Ward et al. [2005] found poor responsiveness to be one of the major causes of time loss and to a lesser extent of user stress in their corpus. Although the authors do not specifically mention the pipeline architecture, responsiveness was harmed both by the endpointing mechanism and the pipeline processing. First, silence detection-based endpointing needs the system to wait for a sufficiently long pause before considering a user utterance as complete. Second, since no information is extracted from the utterance until the end point, all the NLU, DM, NLG and TTS computations have to happen after the user has finished speaking, leading to increased delays.

Two parallel efforts have been conducted to overcome current systems' limitations. First, new architectures have been developed that can support more flexible turn-taking. Second, based on human-human conversation research, explicit models of turn-taking using signals that announce various events such as turn transitions and backchannels, have been built. The rest of this chapter is a review of past work in these two directions.

3.2 Architectures Supporting Flexible Turn-Taking

3.2.1 The Incremental Processing Approach

Incremental Language Understanding

Research in psycholinguistics has established that humans process utterances incrementally [Tyler and Marlsen-Wilson, 1977, Altmann and Steedman, 1988, Kamide et al., 2003]. That is to say, when we hear an utterance, at each point during the utterance, we hold a (potentially underspecified) semantic representation of this utterance. Both in order to match human language processing and to allow interactive natural language-based applications, incremental parsers have been proposed and implemented by computational linguists [Wiren, 1992, Mori et al., 2001, Rose et al., 2002, ACL Workshop on Incremental Parsing, 2004, Kato et al., 2005]. Among those, a number have specifically targetted spoken dialogue systems.

For example, Stoness et al. [2004] describe a continuous understanding module for a dialogue system which takes into account context information (e.g. reference resolution) to guide its parser dynamically. They use a Mediator module which exploits high level information (e.g. from the reference resolution module of a dialogue system) to directly modify the chart of a chart parser, changing the probability of certain chart entries or adding new chart entries. They evaluated their approach on a corpus of human-human dialogues and found that it brought some, albeit quite small, improvement over a non-incremental parser, in terms of understanding accuracy and efficiency.

The incremental parser developped by Nakano et al [Nakano et al., 1999b], uses a unification-based grammar and combines NLU and discourse processing in an algorithm called Incremental Significant-Utterance-Sequence Search (ISSS). For each new incoming word from the ASR, the parser maintains a number of candidate contexts (i.e. parsing stack and beliefs about the user's intention), each associated with a priority level. Priorities are heuristically set so as to be higher for ”significant utterances”, i.e. phrases or clauses that correspond to a fully formed dialogue act, and also favor longer constituents (as opposed to concatenation of shorter ones). This parser is part of the WIT spoken dialogue system toolkit (see below).

Another related work is Wang [2003], which describes an integrated recognition and parsing engine. In this approach, the speech recognizer uses a combination of Probabilistic Context-Free Grammars (PCFG) and N-grams language models to capture the semantic structure of the input (through the PCFG), while keeping the flexibility of N-grams to model the language of complex semantic classes (e.g. ”email subject”). Decoding is then performed frame by frame, as in standard speech recognition, and the current hypotheses, including their semantic PCFG classes, can be obtained at any frame. The method was evaluated in MiPad, a personal assistant for mobile devices that allows the user to manage contacts, emails and schedules. The results show no difference between the synchronous (i.e. incremental) NLU version and a standard ”turn-based” version in terms of task completion rate or time. However, subjects using the incremental NLU version tended to produce longer and more complex utterances since the system showed its current understanding while the user was speaking.

Architectures Built Around Incremental NLU

The TRIPS spoken dialogue architecture [Ferguson and Allen, 1998], developped at the University of Rochester, is an intergrated system for conducting collaborative problem-solving dialogues. It has been used to develop a number of dialogue systems over almost a decade on tasks such as emergency response and evacuation planning [Allen et al., 2000]. While its initial implementation handled turn-taking in the standard rigid way, a later version featured a new core architecture specifically allowing incremental interpretation and generation [Allen et al., 2001]. This new architecture is divided in three components: interpretation, generation and behavior, the latter representing all high level planning. An important feature of TRIPS is the separation of discourse- from task-related components. Discourse is captured by a Discourse Context (DC), which contains typical elements such as the past history of user and system utterances and the set of current salient entities (for reference resolution), but also discourse level obligations (e.g. the obligation to address a question asked by the user), and current turn status. All the components of the architecture run incrementally and asynchronously, which allows for flexible turn-taking behavior. For example, when there is an obligation to respond to the user in the DC, the ”normal” behavior is to get the answer to the user’s question and produce it. However, the generation module might also decide to provide an acknowledgement, possibly in the form of a backchannel, without waiting for the task-reasoning modules’ complete answer. Unfortunately, while Allen et al. [2001] do provide a number of examples of the phenomena that this architecture aims at capturing, there are, to my knowledge, no published studies of a full spoken dialogue system based on it exploiting its turn-taking capabilities. There is, however, recent work [Allen et al., 2005] using a TRIPS-based multimodal system that takes speech input to manipulate graphical objects on the screen, but responses from the system are only graphical.

The WIT spoken dialogue system toolkit was developped at NTT Corporation by Nakano and his colleagues [Nakano et al., 1999a, 2000]. This toolkit provides four main modules. First, a grammar-based speech recognizer outputs word hypotheses as they are identified as being part of the best path in the grammar. Second, the core of the architecture is the incremental natural language understanding module described in section 3.2.1. The third component of the architecture is an incremental natural language generation module described in detail in Dohsaka and Shimazu [1997], which takes into account user responses (including backchannels) to system utterances to dynamically update output planning. Finally, the last module concerns speech output and consists of pre-recorded phrases that are concatenated together according to the NLG output plan. While systems built on the WIT architecture have been used in further research studies (e.g. the work on utterance boundary detection in Sato et al. [2002], see section 3.3) and videos demonstrating such systems have been made public, no formal evaluation of these system has been published.

3.2.2 The Real-Time Processing Approach

Real-Time Control of Mobile Robots

The incremental processing work described in the previous section is rooted in human language processing and psycholinguistics. Another way to approach flexible turn-taking is as an instance of real-time reactive behavior by an artificial agent. This relates to research in Artificial Intelligence and mobile robotics on how autonomous mobile robots can navigate in the real world and perform their tasks. A common principle used to build architectures for such agents is to use multiple layers [Brooks, 1985, Muller, 1996, Hassan et al., 2000]. In such architectures, the components in charge of long-term planning (such as those required to explore a building) and those in charge of immediate reactive behavior (such as those required to avoid obstacles) operate asynchronously and for the most part independently. They communicate usually by passing messages. On the one hand, the reactive component informs the planning component of events arising in the real world that might trigger replanning. On the other hand, the planning component informs the reactive component of its plan of action so that the reactive component can (attempt to) take the prescribed actions. This separation into layers can be ported to spoken dialogue, where the planning component is the dialogue manager and the reactive component (which typically does not exist in standard dialogue systems) is in charge of executing the actions (e.g. saying the utterances) decided by the dialogue manager while monitoring real-world events (e.g. barge-in, backchannel) that might modify the dialogue manager's model of the world.

Spoken Dialogue Architectures based on Real-Time Processing

Aist [1998] describes a turn-taking architecture for the Reading Tutor of CMU's Project LISTEN, a system that listens to children reading aloud and provides help and feedback accordingly. Although the Reading Tutor differs from task-based spoken dialogue systems in that the student spoken input is limited to reading (correctly or not) sentences displayed on the screen, thus eliminating the problem of natural language understanding, it does involve a multimodal ”dialogue” in that, in addition to reading the sentence, the student can click on certain GUI elements (e.g. to request help). On the other side of the interaction, the system uses speech and graphics to provide instructions, feedback, and help to the student. The tutorial actions that the system can take [Aist and Mostow, 1997] include backchanneling when the student is correctly reading the sentence. In order to accomodate the multimodality and allow for time-sensitive actions, the system uses an event-based model, where each event (examples of events include ”user starts speaking”, ”user clicks on the Back button” and ”tutor stops speaking”) is timestamped and certain events trigger transitions in a finite state discourse model. Based on the incoming events and the current state of the interaction, a set of turn-taking rules is used to decide, 5 times per second, whether the system should start/stop speaking or produce a backchannel. Unfortunately, little quantitative evaluation of the turn-taking mechanism of the system has been published (see Aist and Mostow [1999] for a description of the challenges in the evaluation of backchanneling). Although behavior is not explicitly decomposed into planning and reactive components in this architecture (partly because no complex long-term dialogue planning takes place), it does display the ability to monitor real-world events and react to them.

Possibly the most detailed work on reproducing human turn-taking behavior in artificial conversational agents is that of Thorisson [Thorisson, 1996, 2002]. The core of this work is Ymir [Thorisson, 1999], an architecture for conversational humanoids that integrates many aspects of multimodal face-to-face interaction (e.g. hand gestures, gaze, backchannels, but also higher level aspects such as discourse planning) in a unified framework. Although the architecture is presumably task- and domain-independent, all of the work in turn-taking has been done with Gandalf, an embodied agent acting as a guide to the solar system. As in other systems, the architecture contains three major sets of agents: perceptors (and multimodal integrators, which combine information from unimodal perceptors), deciders, which plan the interaction, and behaviors, which encode the realization of communicative actions as motor sequences. In addition, Thorisson divides mental processes involved in conversation in three layers, which affect all three types of modules (perceptors, deciders, and behaviors). These layers correspond to different levels of priority and reactivity and are, from the highest priority to the lowest:

the Reactive Layer which captures the fastest (and simplest) behaviors such as broad-stroke functional analysis and reactive behaviors (e.g. backchannels). Decisions in this layer are made at a frequency of 2-10 per second.

the Process Control Layer which captures domain-independent dialogue structure interpretation and control mechanisms. Such decisions happen 1-2 times per second.

the Content Layer which contains actual linguistic and higher level mechanisms used for content interpretation and presentation. Decisions at this level happen 1 or less times per second.

The argument is that, given the constraints of real-time processing imposed by face-to-face interaction, an agent should first analyze the other participant’s behavior in terms of broad functionality (e.g. is a gesture communicative or non-communicative?), then in terms of dialogue control (e.g. turn-taking), and only finally (i.e. ”when time allows”) in terms of linguistic and discourse content. In practice, deciders and multimodal integrators are coded as sets of rules that get evaluated at a rate depending on their layer. For example, one rule for turn-taking lying in the Reactive Layer is (in LISP-style, reproduced from Thorisson [2002]):

```
START STATE: Other-Has-Turn
END STATE  : I-Take-Turn
CONDITION  : (AND  (Time-Since ’Other-is-presenting > 50 msec)
                   (Other-produced-complete-utterance = T)
                   (Other-is-giving-turn = T)
                   (Other-is-taking-turn = F))
```

This rule, evaluated 2-10 times per second, specifies that the system must take the turn after a 50 ms pause following a user utterance, provided the utterance is complete and turn-yielding. Other examples of rules in the Reactive Layer aim at showing that the system is listening (by modifying Gandalf’s facial expression) or at looking puzzled during an awkward pause. Rules in the Process Control Layer concern higher level behavior such as turning to the user when the system speaks. Gandalf/Ymir was evaluated in three ways: comparison with human behavior, reusability of the framework, and user study [Thorisson, 1996]. First, from a theoretical point of view, Thorisson compared the response times of Ymir’s various perception, decision, and action loops with those of humans and found that they were roughly comparable. This evaluation is however highly dependent on the performance of the hardware and the software used and is probably no longer relevant ten years later. Second, Thorisson showed that his framework is general by using it in two very different systems: Gandalf and Puff the LEGO Magic Dragon, a character in a video game-like environment. More interestingly, he conducted a user study in which 12 subjects interacted with three versions of Gandalf: one with just content (task) related behavior (CONT), one with additional turn-taking abilities (ENV), and one with emotional expressions (e.g. looking confused) but no flexible turn-taking (EMO). In the follow-up questionnaires, subjects rated the ENV system higher than the other two in terms of smoothness of interaction and life-likedness. More surprisingly, they also rated the system's ability to understand and produce natural language significantly higher in the ENV condition than in the other two. One explanation for this would be that the answer to the natural language questions reflected more the overall satisfaction of the subjects than the specific understanding and generation ability of Gandalf.

Most recently, Lemon et al. [2003] proposed another multi-layered architecture for dialogue systems with the goal of separating the planning aspect of dialogue from its reactive aspect. The top layer keeps track of the context of the dialogue and plans responses to user requests and system-initiated topic switches. The bottom layer is in charge of maintaining the communication channel. This involves a variety of tasks such as the generation of outputs scheduled by the top layer, targeted help after non-understandings, and turn-taking (e.g. interrupting the user). None of the actions taken at the bottom layer require information about the context of the dialogue beyond the last system and user utterances. The two layers operate asynchronously and communicate through specific data structures. For example, the Output Agenda is a prioritized list of the outputs planned by the top layer that is accessed by the bottom layer to decide what to generate next. Again, while the authors give examples of behaviors made possible by the architecture and mention a system built on top of it, no formal evaluation results are provided.

3.3 Models of Turn-Taking

Advanced architectures like the ones described in the previous sections are necessary but not sufficient to perform flexible turn-taking. In addition, dialogue systems must be able to decide which turn-taking action to take when, based on a model of turn-taking. To our knowledge, no generic computational model of turn-taking has been proposed to date. However, a number of researchers have modeled signals announcing specific events such as turn transitions and backchannels. After a brief overview of related human-human conversation research, this section reviews previous attempts to build such computational models.

3.3.1 Research in Human-Human Conversation

Conversation Analysis

Pioneered in the 60s and early 70s by Harvey Sacks and his colleagues, Conversation Analysis (CA) is the study of how people interact with speech in different social settings (e.g. informal conversations, medical appointments, political interviews...). In particular, CA researchers have focused on turn-taking as one of the prominent features of spontaneous conversation. Based on an analysis of naturally occurring conversations, Sacks et al. [1974] established a set of constraints that they claim any model of turn-taking in conversation should obey. Namely, that such a model should be locally managed (i.e. only depends on the neighboring turns), party-administered (i.e. does not involve an external regulator), and interactionally controlled (i.e. managed through interaction between the participants). Accordingly, they propose a minimal model whose central elements are Transition Relevance Places (TRPs). TRPs are points in an utterance where it would be *relevant* for another participant to take the turn. To explain the high number of turn transitions without gap (or even with overlap) that they observe, Sacks et al. advance the hypothesis that listeners are able to project TRPs before they occur. Many studies in CA following Sacks et al. [1974] aim at identifying features of TRPs and their projectability.

Sacks et al. [1974] consider syntactic constituents' boundaries as TRPs. This strict use of syntax is problematic considering that spoken language rarely has well-formed complex constituents due to disfluencies (see for example Levelt [1993] for a discussion of these issues). Recently, authors concerned with the quantitative analysis of turn-taking [Ford and Thompson, 1996, Furo, 2001] have given more operational definitions of syntactic TRPs. For instance, Ford and Thompson [1996] specify that they are ”potential terminal boundaries for a recoverable *clause-so-far*”. This is illustrated in the following example (from Ford and Thompson [1996]):

(3.1) V: And his knee was being worn/- okay/ wait./
It was bent/ that way/

In this utterance, the two syntactic completion points in ”It was bent/ that way/” indicate that at these two points the whole units ”It was bent” and ”It was bent that way” can be considered complete. Often, a larger context must be taken into account to recover the clause-so-far. For instance, answers, which are often not complete syntactic constituents in spontaneous conversations, complete a clause *implied* by the question, and are therefore considered complete syntactic units from the point of view of turn-taking. Similarly, Reactive Tokens such as backchannels (e.g. ”okay”, ”mhm”), assessments (e.g. ”really?”) and repetitions (when the listener repeats all or part of the original speaker's utterance as an acknowledgment or confirmation) are often considered complete syntactic units although they are typically not well-formed syntactic constituents [Ford and Thompson, 1996, Sorjonen, 1996, Furo, 2001].

A second type of TRP features is prosody. In English, falling and rising pitches are usually considered as markers of intonation completion points for statements and questions, respectively [Chafe, 1992, Ford and Thompson, 1996, Furo, 2001]. In contrast, level pitch is considered a continuation marker, therefore not indicative of a TRP [Oreström, 1983, p. 62]. Another prosodic pattern is the lengthening, or drawling, of the final word or syllable of a turn [Duncan, 1972, Koiso et al., 1998], although these results are contradicted by Oreström [1983, p. 64]. These phenomena are sometimes accompanied by changes in voice quality such as creaks [Chafe, 1992].

Semantic and pragmatic completion points correspond to points where the turn-so-far constitutes a complete meaningful utterance coherent within the conversational context. Not surprisingly, they have been found to strongly correlate with TRPs [Oreström, 1983, Furo, 2001]. However, all the authors addressing this point acknowledge the difficulty of establishing an operational definition of semantic completion. Hence, Oreström [1983, p. 57] writes:

> As there is no simple way to formalizing a semantic analysis of this conversational material, the interpretation will have to be made subjectively. Admittedly, this may be regarded as the weakest point in the whole analysis.

Furo [2001] uses a more specific definition of semantic completion points. They are placed after: floor-giving utterances (e.g. questions), complete propositions, and reactive tokens, provided they are not preceded by a floor-claiming utterance (such as ”Here is what I told him.”). However, there is still an issue with this criterion since it relies on syntax completion.

Finally, non-verbal aspects of face-to-face conversation are known to interact with linguistic signals for turn-taking. The most frequently observed such signal is that of speakers making eye contact with their listeners to indicate the end of their turn [Kendon, 1967, Goodwin, 1981]. On the other hand, gestures are generally found to be of little turn-taking significance [Oreström, 1983, p. 36], although Duncan [1972] did find that hand gestures were used by patients and doctors during interviews to cancel other turn-yielding signals. Beattie [1983] makes a distinction between simple movements, through which the speaker merely emphasizes the spoken message, and more complex gestures, which appear to accompany the planning of an upcoming utterance, the gesture generally preceding the verbal rendition of the utterance. While one can speculate about listeners using these gestures to predict upcoming speech (which would go along with Duncan's finding that gestures signal turn continuations rather than transitions), more studies are necessary in this area to expose the potential turn-taking functions of gestures.

Psycholinguistics

Psycholinguistic studies of turn-taking differ from Conversation Analysis research in that they focus on perceptual processes. Usually, psycholinguistic experiments involve recordings from natural conversation or read speech, that are sometimes manipulated to remove certain features (e.g. lexical). For example, in an experiment on turn-taking in Dutch conversations, Wesseling and van Son [2005] asked their subjects to listen to recordings of natural conversations in two conditions: natural speech and synthesized speech, the latter preserving the prosody of the natural utterances but stripping them away of any lexical/semantic information. Subjects had to produce minimal vocal answers (backchannels) at times they felt were appropriate, which, the authors claim, would correspond to TRPs. The delay between each minimal response and the closest (manually labelled) turn boundary was then measured. Their results show that human listeners are consistently able to react very fast to, or even anticipate, turn boundaries both with full speech and with only prosodic information. This result is in partial contradiction with a previous study by Schaffer [1983], who found little consistency in how listeners use intonation to predict TRPs. In her experiment, subjects were listening to short extracts (”sentences, phrases, or single words”) from recorded natural conversations in again two similar conditions: normal recording and filtered. Based on what they heard, on one set of recordings, the subjects had to predict whether the next speaker in the original conversation (immediately after the extract) was the same as the one in the extract or a different one. Schaffer then analyzed whether groups of 20 to 31 listeners agreed in their judgements. She found only sparse evidence of agreement between the subjects. In particular, intonation alone (in the filtered condition) did not allow them to agree on points of speaker transition. Although part of the results might be explained by her experimental design (e.g. quality of the recordings, unnaturalness of the task, fact that speaker changes are often optional in natural conversation, etc), they do support the claim that intonation *alone* and *without context* is not a sufficient cue for turn-taking.

3.3.2 Utterance Endpointing

Except for systems that use a push-to-talk interface, all dialogue systems face the problem of detecting the end of user utterances. In most cases, simple pause detection, i.e. an algorithm that detects the presence of speech in the audio signal, is used. Nevertheless, even the apparently trivial task of discriminating speech from non-speech in the audio signal is harder than what one might expect, in particular in noisy environments. Some researchers have therefore studied robust speech detection (e.g. for in-car systems see Hariharan et al. [2001]). Obviously, this way of relying purely on pauses to segment speech does not correspond to what humans do when they take turns in a conversation. Indeed the work described in section 3.3.1 shows that: 1) there are many signals *before* the speaker finishes her turn that announce the upcoming end of this turn, and 2) human listeners make extensive use of those signals to decide when to start speaking. Besides the somewhat abstract issue of how well dialogue systems mimic humans, the complete absence of projection in endpointing creates unnecessary delays in system responses. One specific problem is that current endpointers do not distinguish between pauses within utterances due to hesitations and floor-yielding pauses at the end of an utterance. In an attempt to address these issues, Ferrer and her colleagues [Ferrer et al., 2002, 2003] use prosodic and lexical information to anticipate the end of user turns. Their method successfully reduced the delays by as much as 81% (when using both lexical and prosodic information) or 64% (when using only prosodic information, thus avoiding relying on speech recognition) compared to a standard pause-based baseline. On the other hand, Bell et al. [2001] rely on semantic and lexical cues to classify pauses as ”closing” (i.e. the system can take the floor) and ”non-closing”, in a real-estate information dialogue system. The lexical cues are words that normally do not appear at the end of an utterance, such as ”or”, ”and”, ”no”, ”a”. The semantic cues are based on system- and state-specific hand-written rules such as ”If the system has described an apartment in the previous turn, e.g. ’The apartment on Kungsgatan has a bathtub’, then elliptical fragments such as ’And the apartment on Hagagatan?’ should be considered closing.” Unfortunately, the authors do not provide any evaluation of the accuracy of the system. Finally, Sato et al. [2002] describe a method based on decision trees to classify pauses of 750ms or more into final and non-final in user interaction with a Japanese toy spoken dialogue system. They use a wide range of features, from prosodic (pitch and power) to lexical (keywords) to semantic (slots filled in the user utterance) cues. Their analysis shows that only lexical and semantic cues help classification. The tree that they obtain is also very task-specific. Therefore, they conclude that they need more or better features (including prosodic features) to create a general model of turn-taking that can be reused across domains.

3.3.3 Backchannel Feedback

In additoin to the signals announcing the end of a turn there are other turn-taking signals, in particular those that trigger backchannel feedback from the listener. Clancy et al. [1996] investigated the use of Reactive Tokens in English, Japanese, and Mandarin. They found that, in English and Mandarin, RTs are produced by the listener predominantly at syntactic completion points (resp. 78% and 88% of the RTs) whereas only 36% of RTs in Japanese conversation occur at such points. This, along with the fact that backchannels are much more frequent in Japanese (29.9% of all speaker changes are backchannels) than in English (15.9%) and Mandarin (4.7%), reflect the typical behavior in Japanese to produce backchannel after each noun phrase produced by the speaker (not only clauses). In her own work, Furo [2001] found a similar percentage of RTs occurring at syntactic completion points in English conversations (71.9%) but a much higher rate in Japanese conversations (83.7%). She also found that many RTs occur at noticeable pauses in both English and Japanese (resp. 74.6% and 82.1%). Duncan and Fiske [1985] hypothesized that, in face-to-face interaction, the *speaker within-turn signal*, which triggers a backchannel from the listener, is composed of the completion of a syntactic clause and of a shift of the speaker's gaze towards the listener.

Ward and Tsukahara [2000] studied another potential backchannel trigger, again in English and Japanese. Their hypothesis was that regions of low pitch yield backchannel feedback. They built a predictive rule for each language and automatically predicted backchannels in two corpora, one consisting of 68 minutes of conversations between native speakers of English, and one of 80 minutes of conversations between Japanese speakers. Although the accuracy of their predictions is modest (18% for English, 34% for Japanese), it is above chance and shows that some of the backchannels can be explained by such a prosodic cue. Koiso et al. [1998] also used predictive models as an analysis tool and trained decision trees to predict backchannel occurrence in Japanese conversations. The recordings were automatically split into ”interpausal units” (stretches of speech separated by pauses of 100 ms or more). For each unit, a number of syntactic and (discretized) prosodic features was semi-automatically extracted. The authors first studied the correlation between each feature and the occurrence of a backchannel in the pause following the unit and then built a decision tree using all the features to predict the backchannels. As a result, they are able to predict backchannel occurrence with a high accuracy (88.6%). They found that intonational features (the type of final contour of the unit) are the strongest signals for the occurrence of backchannels. By contrast, the most prominant features inhibiting the occurrence of backchannels are syntactic (i.e. certain types of words, such as adverbs, tend to ”block” backchannels). In her analysis of videotaped conversations between strangers, Denny [1985] built logistic regression models to predict the occurrence of backchannels during pauses of 65 ms or more. Although she does not provide an estimate of the quality of the prediction, she did find that the most useful (hand-annotated) features were grammatical completion, speaker gaze, intonation, pause length, and type of preceding utterance (questions are less likely to be followed by backchannels than statements). Cathcart et al. [2003] attempted to predict backchannel feedback in the HCRC Map Task Corpus of goal-directed conversations using low-level, easy to extract features. By combining pause duration information with a trigram language model built on symbols representing Parts-of-Speech and pauses from the main speaker, as well as backchannels from the listener, they were able to achieve precisions in the range of 25%-30% and recalls of 35%-60%.

As we see, the literature on cues triggering backchannel feedback indicates that somewhat similar features are used for backchannel and turn transitions. One difference is that backchannel feedback is always optional (particularly in English), whereas turn transitions are more systematic. Indeed, one can imagine a completely functional conversation without any backchannel, but not without turn transitions. Therefore, it is difficult to evaluate the acuracy of backchannel predictions, since they attempt to detect places where a backchannel feedback *could*, but does not have to, be produced. As for turn transitions, it seems that one weakness of all the analyses described above is that, to a large extent, they ignore semantic and pragmatic information, which might be of prime importance in the decision to produce backchannel feedback (e.g. the fact that a speaker sounds sad or concerned might lead a caring listener to produce backchannels, as a form of support).

Chapter 4

An Architecture for Handling Turn-Taking in Spoken Dialogue Systems

4.1 The Olympus Dialogue System Architecture

4.1.1 Overview

The Olympus architecture is a set of programs which, put together, constitute a complete spoken dialogue system. Olympus provides a clear separation between task-independent ”engines” and task-dependent ”resources”, which facilitates the construction of new dialogue systems. Although more complex than off-the-shelf toolkits such as Nuance’s OpenSpeech Dialog[Nuance], Olympus is also more powerful and is aimed at supporting research in the various areas of spoken dialogue systems. For example, it allows a researcher to build a full system quickly and then focus on different natural language understanding techniques.

Historically, Olympus finds its roots in the CMU Communicator project [Rudnicky et al., 2000], which resulted in the creation of a dialogue system for travel planning, using a novel, agenda-based dialogue manager [Rudnicky and Xu, 1999]. CMU Communicator used the Sphinx 2 speech recognizer [Huang et al., 1992] along with Phoenix, a robust parser for spoken language understanding [Ward and Issar, 1994]. On the generation side, Rosetta, a template-based generation toolkit was created, although stochastic language generation has also been investigated [Oh and Rudnicky, 2000]. Finally, high-quality speech synthesis was provided by a limited-domain voice on the Festival speech synthesis system [Black and Lenzo, 2000]. All these modules communicated using a Galaxy-II hub [Seneff et al., 1998]. Building upon this experience, RavenClaw [Bohus and Rudnicky, 2003], a task-independent dialogue management framework, was designed and implemented within the Galaxy framework. As more systems were built using RavenClaw, all the modules of the original architecture were refined and improved, while still kept task-independent. Currently, several research projects at CMU use the Olympus architecture as the base for their system, including Let's Go [Raux et al., 2003, 2005], RoomLine [Bohus and Rudnicky, 2005a,b] and TeamTalk [Harris et al., 2005].

The rest of this section describes the ASR/NLU modules and the NLG/TTS modules, while the core modules Interaction Manager and Dialogue Manager are dealt with in the next sections.

4.1.2 Speech Recognition and Understanding

The incoming audio signal is first processed by a module called the audio server. Its function is to capture the input from the microphone or telephony board and send it to a set of parallel speech recognition engines. Typically, there are two Sphinx 2 [Huang et al., 1992] engines that use different acoustic models: one was trained on male speech and the other on female speech. However, other combinations of two or more engines could be used (e.g. one with a generic statistical language model and another with a state-specific finite-state grammar). For telephone-based applications, a DTMF (touch-tone frequency) decoding module has also been implemented. Its interface is similar to the other recognition engines.

To detect utterances' start and end points, the audio server can use its own adaptive energy-based endpointing mechanism or it can receive instructions on when to endpoint from external modules. As we will see, the latter mode is the default one for Olympus v2, where the Interaction Manager (see section 4.3.3) is in charge of endpointing. Within an utterance, the audio server streams the audio signal to the engines, reads partial recognition results from them, and sends those results to the hub. Once an utterance has been endpointed, the audio server requests a final hypotheses from all the engines and sends them to the hub. The framework also allows the use of N-best lists although, for recognizer performance reasons, live systems currently only use the first hypothesis.

Once recognition hypotheses have been obtained, they are sent to the Phoenix semantic parser [Ward and Issar, 1994], which builds a semantic frame for each hypothesis. Phoenix uses hand-written CFG-like grammars and is robust to ungrammatical input (e.g. out-of-grammar words and fragmented sentences). In some ambiguous cases, it will output several possible parses for a single hypothesis. All parses of all hypotheses are gathered and sent to the hub.

The last component of the natural language understanding side of Olympus is Helios [Bohus and Rudnicky, 2003], a confidence annotator. Helios takes the set of parses output by Phoenix and computes additional features for them (e.g. utterance length, parse coverage, etc). In particular, it uses a logistic regression model trained on data labeled with concept errors to compute a semantic confidence score for each hypothesis. This score reflects the probability that the hypothesis contains one or more concept substitution, insertion, or deletion. Finally, Helios selects the hypothesis/parse with the highest confidence score and sends it to the hub.

4.1.3 Speech Generation and Synthesis

On the output side, Olympus uses a template-based natural language generation module called Rosetta. Written in Perl, Rosetta maps semantic frames output by the DM to English sentences.

The output of Rosetta is passed through the hub to Kalliope, which acts as an interface to speech synthesizers. Currently, two synthesizers are supported: Festival, using a separate synthesis server accessed through a socket connection, and Cepstral's Swift[1], using the API to synthesize locally. Kalliope is in charge of obtaining the wave form for the given output and of sending it back to the hub. In Olympus v1, Kalliope was also in charge of playing the synthesized outputs in the order in which they came and stop playing in case of user barge-ins. In the new version, these functions have been transferred to the Interaction Manager and Rosetta simply returns the synthesized waveform to the IM via the hub.

4.1.4 The RavenClaw Dialogue Manager

RavenClaw[Bohus and Rudnicky, 2003] belongs to the family of plan-based dialogue managers. It uses a tree to capture the static[2] structure of the task. The nodes of the tree are Dialogue Agents that can be executed to perform part of the dialogue. More specifically,

[1] http://www.cepstral.com

[2] While RavenClaw does allow to dynamically expand the tree at runtime, we do not use this feature in this work.

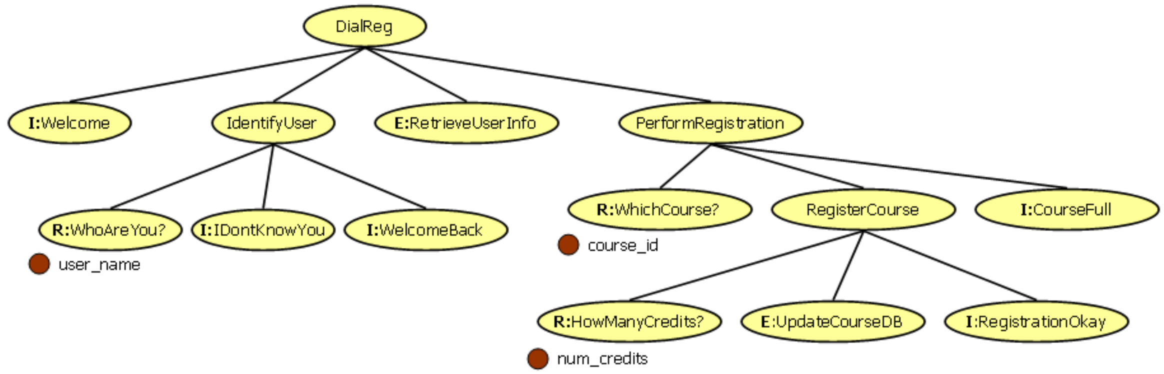


Figure 4.1: Example RavenClaw Task Tree

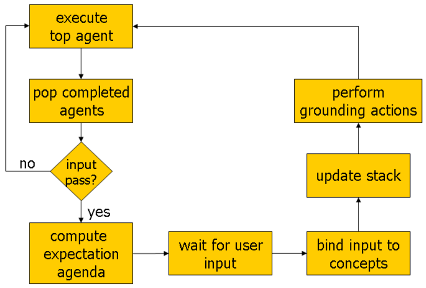
non-terminal nodes (or Agencies) manage sub-dialogues by scheduling the execution of their children. Terminal nodes are of four types:

- **Inform Agents** convey information to the user (e.g. speaking a statement)
- **Request Agents** obtain information from the user (e.g. asking a question)
- **Expect Agents** capture information from the user, without explicitly requesting it
- **Execute Agents** perform internal operations and access backend components (e.g. a database)

Figure 4.1 shows a simple task tree. The tree is all the dialogue designer has to provide in order to build the dialogue manager of a new system. RavenClaw uses the information in the tree to interpret user utterances, maintain the state of the dialogue, and decide which actions to take. At any time, the state of the dialogue is described by the following elements:

The Dialogue Stack describes the current focus of the dialogue. When a new topic is started, the corresponding agent (or agency) is pushed on top of the stack. When a topic is closed, it is popped from the stack. Thus the stack contains not only the topic under immediate discussion, but also the previous topics that have not yet been closed.

The Expectation Agenda is built from the stack and describes what the user is expected to say at this particular point in the dialogue. To assemble the agenda, RavenClaw



1	DialReg	
2	Welcome,DialReg	S: Welcome to DialReg!
3	DialReg	
4	IdentifyUser,DialReg	
5	WhoAreYou,IdentifyUser,DialReg	S: Who are you?
	[user_name],[course_id],[num_credits]	U: John Doe
6	IdentifyUser,DialReg	
7	WelcomeBack,IdentifyUser,DialReg	S: Hi John!

Figure 4.2: Flowchart of the core loop of RavenClaw v1 and example execution of the tree from Figure 4.1. Each row in the table represents the execution of one agent, except for the grayed row which corresponds to a user input. The middle column gives the stack (with the top agent at the left end), except for the grayed row where it represents the agenda. The rightmost column gives the system (S:) or user (U:) utterance corresponding to the execution/input pass.

traverses the stack from top to bottom and obtains the list of concept expected by each agent. In the example of Figure 4.1, the WhereAreYou Request Agent expects the `user_name` concept, whereas the PerformTask Agency expects `course_id` and `num_credits`, since they are both attached to agents from its sub-tree. The ordering of expected concepts in the agenda, from the most relevant topic (on top of the stack) to less relevant ones, allows the system to understand user input in the right context, while still allowing the user to switch topic if desired.

The Task-Specific Concepts represent the information acquired from the user so far. Each concept's value is a set of competing hypotheses that were obtained in the course of the dialogue, potentially through multiple speech understanding results. RavenClaw's integrated grounding mechanism is in charge of deciding (through dialogue) which hypothesis reflects the actual user intention.

Based on these elements, we present two algorithms to execute a dialogue. The first one, which we call Turn-Synchronous Dialogue Management, was implemented in RavenClaw v1. The second one, named Event-Driven Dialogue Management, is the new approach we have implemented in RavenClaw v2.

4.2 RavenClaw v1: Turn-Synchronous Dialogue Management

4.2.1 The Core Loop of RavenClaw v1

At runtime, the core loop of the RavenClaw v1 engine proceeds as follows. The dialogue stack initially contains the root of the tree. Each of the subsequent iterations is decomposed in two phases:

The Execution Phase during which agents are put on the stack (by other agents, typically agencies that schedule the execution of their children) and the engine executes the agent on top of the stack,

The Input Phase triggered by Request agents, during which the system constructs the expectation agenda according to the procedure described in section 4.1.4, waits for an input from the user, and interprets it according to the agenda. This phase might also add agents to the stack to perform grounding actions and because agents from the task tree can be ”triggered” (i.e. pushed on the stack) by concepts in user inputs (e.g. any time the user says ”Help”, an appropriate agent giving context-sensitive help is pushed on the stack).

Figure 4.2 shows a flowchart of the main loop of the RavenClaw v1 engine, along with an example execution based on the tree from Figure 4.1. An important point is that Execution Phases (e.g. steps 1-5) are performed and asynchronously from the actual conversation. Thus, in this example, the execution of step 2 results in the DM sending the output frame to the NLG, immediately popping the Welcome Inform Agent from the stack and pursuing the execution. This asynchrony allows the system to generate and synthesize its prompts in advance, while it is speaking previous prompts. In our example, the prompt from step 5 is synthesized just as the prompt from step 2 is being played. This reduces delays between system prompts, and is thus important to allow proper real-time execution. On the other hand, the Input Phase (the grayed row in Figure 4.2) *assumes* that, when the user input reaches it, the dialogue is indeed in the state corresponding to the Request agent that triggered it (at step 5), i.e. that the user spoke after hearing all the scheduled prompts. Because of the fact that the dialogue manager relies on synchronicity between the real world and the internal dialogue state at each Input Phase, we call this approach Turn-Synchronous Dialogue Management (TSDM). We will see in the next sections that, in practice, TSDM can lead to misinterpretations and turn-taking failures.

4.2.2 Interpretation and Turn-Taking Issues in TSDM

Problems arise with TSDM when the user speaks during system prompts, either to barge-in or as a backchannel. For instance, let us assume that the system sends an implicit confirmation prompt (”A room for next Tuesday.”) immediately followed by a question (”Do you want a room in Wean Hall?”). If the user responds to the implicit confirmation prompt by saying ”yes” immediately after it, the system will interpret it as an answer to the following question, even though the user has not heard it. This is particularly problematic because the user usually does not realize that there was a misunderstanding. Therefore the dialogue continues on ”misgrounded” bases. In addition, two turn-taking problems can arise in this situation. First, even before the interpretation issue, the fact that the user’s ”yes” to an implicit confirmation prompt is taken as a barge-in, resulting in the system interrupting itself. Ideally, the system should interpret the backchannel as an indicator that the confirmed concept was indeed correct, but still continue to speak the rest of its utterance. This is not allowed by TSDM since user input is always taken as a turn. Second, because it does not monitor exactly when user utterances are produced, TSDM allows turn overtaking (see section 3.1). If, for example, an utterance is split in two by the endpointer, the second half will be interpreted as an answer to a question that was not yet asked. This leads to great confusion on the part of the user.

4.3 RavenClaw v2: Event-Driven Dialogue Management

4.3.1 Events and Actions

A first modification in RavenClaw v2 has been to generalize the concepts of input and output as events and actions. Events represent anything happening in the real world that is relevant to the dialogue. This definition includes user inputs but also partial utterances, notification of system utterances (i.e. when they are actually said to the user), and even non-conversational events (e.g. a robot encountering an obstacle in a robot control system). One additional feature of events is that they are all contextualized. That is to say, each event reaching the DM is labeled with an index pointing to the dialogue state in which it occurred. Symmetrically, actions represent anything through which the system acts upon the real world. This includes system prompts but also gestures (for an embodied agent) and non-conversational actions (e.g. moving a robot). Currently, actions are always executed in the order they were issued by the DM. However, ultimately, each action will be tagged with a priority level, as well as a relevant time-period for optional actions. Actions should be executed in order of decreasing priority, and actions that are not yet executed but no

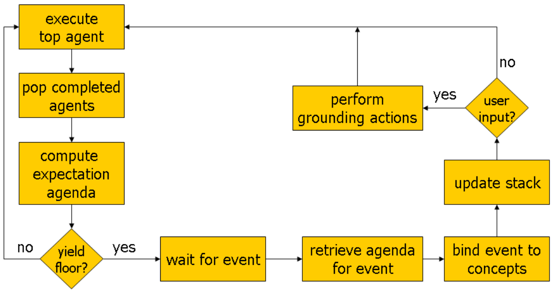


Figure 4.3: Flowchart of the core loop of RavenClaw v2

longer relevant are canceled (the DM being notified of the cancellation). This latter feature will allow for example the DM to request a backchannel as a response to a particular user utterance, and, if the backchannel could not be produced in time (e.g. because the user never paused before introducing new content), it is canceled.

4.3.2 The Core Loop of RavenClaw v2

The core loop of RavenClaw v2 is shown in Figure 4.3. One crucial difference with RavenClaw v1 is that the possession of the conversational floor is now explicitly represented. Initially, the system is assumed to have the floor. Then, after the execution of a request agent, the system yields the floor to the user, until a complete user utterance reaches the DM, at which point the system gets the floor back. The user can also grab the floor by barging in on the system. When the system does not have the floor, agents, by default, cannot be executed (i.e. the system just skips the execution part of the core loop and waits for the next event). Certain agents, however, can be declared as not requiring the floor, in which case they can be executed at anytime if they are on top of the stack (e.g. agents handling backchannel behavior do not require the floor). The other differences between the core loops of RavenClaw v1 and v2 are as follows:

1. The agenda is computed after every agent execution, instead of only at Input Passes.
2. When Request Agents yield the floor to the user, the system waits for any event, and interprets it according to the agenda of the dialogue state in which it actually happened (i.e. the state that the event is marked with).
3. Only if the event is a user utterance are grounding actions taken.

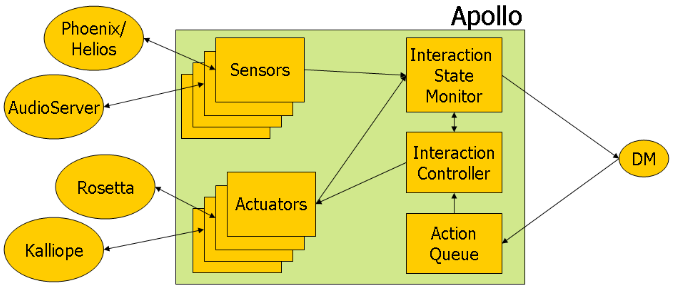


Figure 4.4: Architecture of the Apollo Interaction Manager

4.3.3 Apollo: the Interaction Manager

Overview

While RavenClaw v2 ensures that events are interpreted in the right context and eliminates the requirement for rigid turn-taking, it does not in itself improve the system's turn-taking behavior. Turn-taking is explicitly handled by a new module called the Interaction Manager (IM), whose current implementation is Apollo.

Broadly speaking, the IM acts as the interface between the real world and the symbolic representation of dialogue within the DM. It monitors events, identifies the dialogue state in which they occur and sends the relevant ones to the DM. On the other hand, it receives action requests from the DM and performs these actions in the right way, at the right time. In particular, it is in charge of keeping the interaction smooth by handling interruptions, backchannels, and smooth turn-taking in general.

Implementation

The general architecture of Apollo is shown in Figure 4.4. There are five components:

- a set of sensors, which connect the IM to external modules such as ASR and NLU. They capture and store events as they occur. Examples of events include a syntactic boundary (from a partial utterance), an intonation boundary, or current pause/speech information.
- an interaction state monitor, which computes state variables from the sensors. Whereas sensors are asynchronous and can receive events at any time, the interaction state combines information from the sensors in a synchronous way. The state variables are defined at any point in time.

- an action queue, which contains the pending action requests received from the DM, in order of decreasing priority.
- an interaction controller, which runs the main loop of the IM. At each iteration (approx. every 100 ms), it asks the interaction state monitor to update the current state and selects the next action to take, based on the state and the action queue.
- a set of actuators, which execute the actions decided by the interaction controller.

Communication between the components is done by event broadcasting. Sensors can register to receive certain types of events when these events either reach the IM from an external module, or are generated by the interaction controller. Similarly, actuators can register for actions, which are broadcasted by the interaction controller. In addition to receiving events from external modules, sensors can configure these modules dynamically, based on broadcasted events. For example, for state-specific language model-based speech recognition, the ASR sensor can send the new LM state to the ASR when the interaction controller decides to start saying a new prompt (thus advancing the state of the dialogue). Currently, the Interaction Controller uses a set of hand-crafted rules to select actions. In the future, a data-driven action selection model will be built, as described in Chapter 5.

4.4 Summary: Work Items

The goal of this part of the work is to provide an architecture to support the research work described in the next two sections. The work items involved and their current status are described in the following table.

Completed	Designing an architecture (Olympus v2) for real-time event handling in spoken dialogue systems
Completed	Modifying the existing ASR/NLU and NLG/TTS components to work in Olympus v2
Mostly completed	Writing RavenClaw v2
Completed	Implement a preliminary version of the Interaction Manager (Apollo) with hand-written rules
In progress	Test the architecture

Chapter 5

Modeling Turn-Taking in Task-Oriented Dialogues

5.1 Introduction

So far, most attempts at modeling turn-taking computationally have been based on handwritten rules (see section 3.2). The problem with such models is that they require expert knowledge on turn-taking to build and don't lend themselves to learning and adaptation. The rules are usually written with the average human behavior in mind although individual differences due to gender, culture, and personality are known to significantly affect turn-taking [Beattie, 1982]. Therefore, the system's behavior might be good on average, without being optimal for anyone, and potentially harmful to certain classes of users. To address this issue, we propose to design a more flexible model based on a reactive action selection mechanism such as the ones that have been studies for years in mobile robotics. Such models have the potential to be optimized on data or even adapt to each user.

5.2 A Computational Model of Turn-Taking Behavior

5.2.1 Reinforcement Learning for Turn-Taking Action Selection

In the proposed work, we will cast turn-taking as the problem of dynamically selecting appropriate actions based on observations from sensors. This problem has been investigated by many researchers in robotics, AI, and cognitive science [Maes and Brooks, 1990, Blumberg, 1994, Andronache and Scheutz, 2002]. One common approach to learning action selection policies is Reinforcement Learning (RL) [Humphrys, 1996]. RL is a form of machine learning where an agent, as it interacts with its environment, receives rewards (or feedback) for the actions it takes. Based on the received reward, it learns what states of the world have more value (i.e. yield positive reward) and which actions lead to valuable states. This approach differs from *supervised learning* in that the system is not given what the ”correct” action is at each point. Instead, the agent learns by ”trial-and-error”. This approach suits the turn-taking problem well because, in many cases, we do not know what the correct turn-taking action is, or even if there is such an action. For example, the research on backchannels reviewed in section 3.3.3 showed that it is indeed very difficult to reliably and consistently label when backchannel should or should not be produced. Even if we were able to get reliable labels, there is no evidence that what we would learn from human-human data would transfer to the human-computer situation.

However, as we have seen in section 2.3, it is possible to identify when ”things go wrong” and ”things go well” on existing human-computer data. RL provides a framework to select actions so as to minimize turn-taking failures and maximize the efficiency of communication. Formally, RL requires the definition of three components:

- a **state space** that is an abstraction of the state of the world at any point. It has to contain all the variables that are relevant to our problem (i.e. turn-taking)
- a set of **actions** that the system can take at any point
- a **reward function** that captures the ”quality” of each state

As much as possible, all of these should be defined in a task-independent fashion, so that we can use the learnt turn-taking model on different systems. We now define these three components in more details for the turn-taking problem.

Turn-Taking Actions

The turn-taking actions are defined in Table 5.1. Note that these are channel-level actions, which do not use knowledge at the task (e.g. whether the system needs to ask a specific task-related question) or grounding (e.g. whether the system needs the user to confirm the value of a specific concept). Practically, this means that, for example, the system can decide to take the floor even though there is no prompt in the prompt queue. In this case, the system will produce a filler, so as to grab the floor in anticipation of an upcoming task-level prompt.

Action	Description
LISTEN	The system remains silent
TAKE_FLOOR	The system attempts to grab the floor by starting to speak
HOLD_FLOOR	The system keeps speaking its current turn
YIELD_FLOOR	The system stops speaking and leaves the floor to the user
BACKCHANNEL	The system speaks without attempting to take the floor

Table 5.1: Turn-taking actions

As is often the case in RL, some actions are only possible in certain states. For example, ”LISTEN”, ”TAKE_FLOOR”, and ”BACKCHANNEL” can only happen when the system does not already have the floor. Conversely, ”HOLD_FLOOR” and ”YIELD_FLOOR” are only valid actions when the system has the floor. These constraints reduce the size of the (state,action) space, hence making the learning task easier.

Turn-Taking States

The definition of the state space is a crucial aspect of RL. On the one hand, states that are defined too generally (i.e. at a high level of abstraction) might fail to capture important signals conditioning the success of different actions. On the other hand, defining states too specifically leads to an explosion of the state space, which, in turn, leads to data sparseness issues. The process of grouping specific states together into more general (or abstract) states is called state aggregation. Another issue is that traditional RL assumes a discrete state space. Several techniques have been proposed to deal with continuous states such as state clustering and function approximation. Usually these methods are generalized forms of state aggregation. In the following, we will describe state variables at a very low level of abstraction (with potentially continuous values), given that when actually building the model, we will use state aggregation and approximation techniques to make the learning task tractable.

State variables that will be investigated are described in Table 5.2. System variables are all known, based on information from the TTS and NLG modules. User variables have to be inferred from the incoming audio signal. ”speaking” and ”turn_duration” will be obtained using a voice activity detector. ”f0_boundary” will be estimated using a pitch extractor combined with a pattern classifier. ”syntax_boundary” will be based on the parse of the current partial ASR hypothesis (if a hypothesis is fully parsable, it ends at a syntax boundary). Finally, ”semantic_boundary” indicates how expected the user input is in the current dialogue state, based on RavenClaw’s Expectation Agenda.

Variable		Values	Description
System	speaking	bool	is the system currently speaking?
	f0_boundary	bool, N/A	is the system at an F0 boundary?
	turn_duration	int, N/A	how long has the current turn lasted?
	pause_duration	int, N/A	how long has the current pause lasted?
	syntax_boundary	bool, N/A	is the system at a syntax boundary?
	semantic_boundary	bool, N/A	is the system at a semantic boundary?
User	speaking	bool	is the user currently speaking?
	f0_boundary	bool, N/A	is the user at an F0 boundary?
	turn_duration	int, N/A	how long has the current turn lasted?
	pause_duration	int, N/A	how long has the current pause lasted?
	syntax_boundary	bool, N/A	is the user at a syntax boundary?
	semantic_boundary	bool, N/A	is the user at a semantic boundary?
Task	discourse_obligation	USER, SYSTEM	who has an obligation to speak?
	prompt_priority	bool, N/A	what is the priority level of the next prompt?
	backchannel_pending	bool	is there a backchannel prompt in the queue?

Table 5.2: Variables describing the turn-taking state

”discourse_obligation” indicates which participant currently has an obligation to speak. This is determined by a stack of obligations constructed as follows. When the dialogue starts, the stack contains ”system”. Then, obligations are pushed on and popped from the stack according to the dialogue acts of the two participants. Each time a participant asks a question or makes a request, the other participant is pushed on top of the stack. Each time a participant answers to a question or a request, they are popped from the stack. This definition comes from the observation that, in task oriented dialogues, there is usually a participant who is in charge of keeping the conversation going. This can be either because they have the initiative (i.e. they initiated the current subdialogue) or because they have to answer a question/request (see Allen et al. [2001] for a discussion of discourse obligations). ”prompt_priority” indicates the level of priority of the top prompt in the IM’s prompt stack (see section 4.3.1). It contains ”N/A” if the stack is empty. ”backchannel” is a flag indicating whether there is a pending backchannel prompt in the stack.

Reward Function

The goal of the reward function is to provide both immediate and delayed feedback from the real world on how well the system is performing. There are many possible rewards related to turn-taking actions.

A first, straightforward approach is to use human labels of turn-taking failures, such as the ones defined in section 2.3. Negative reward is given when a turn-taking failure happens. This requires the definition of scheme (based on the preliminary scheme of section 2.3 that can be applied consistently and reliably to the data. We will investigate such a scheme as part of our work on evaluation (see chapter 6).

A more objective (negative) reward function can be defined as the duration of gaps and overlaps between the utterances of the user and the system (or simply a negative reward for each time step when there is overlap or gap). This directly follows from the hypothesis of Sacks et al. [1974] that human turn-taking behavior aims at minimizing gaps and overlaps in conversation. This function has the advantage that is possible to compute it automatically with high accuracy (simply based on voice activity detection). It could therefore be used in an online learning setting, where the system improves its turn-taking behavior over time without human intervention.

The reward function could also relate turn-taking with understanding. For example, a negative reward corresponding to word error rate or concept error rate could be given for each user input. This requires at least transcription of the data, potentially concept error annotation. An unsupervised version of this would use the confidence value of the user input as a reward. Again, the advantage is that this could work online. Of course, the accuracy of the reward would be highly dependent on the confidence annotator used.

Finally, there might be a need to explicitly give a reward for backchannels (both from the system and the user). This reward should translate the grounding function of backchannels and reflect the additional information they provide in terms of confidence over the value of concepts. This reward is particularly important if a negative reward is given to overalpping speech since that would most likely result in avoiding backchanneling at all times.

We will evaluate the different behaviors yielded by different reward functions. In addition, we can combine the rewards defined above in a single function. The problem is then to determine the weight for each type of reward. This can be done either heuristically or by empirically estimating the impact of each feature on the overall dialogue.

5.2.2 Alternative Approach: Supervised Learning

Although, as explained above, RL has many advantages for learning turn-taking behavior, there are also potential issues. In particular, RL requires a large amount of data since all states have to be visited enough time to build reliable estimates of their value. If, after initial studies with RL, we come to the conclusion that it cannot be applied to this problem, we will back off to supervised learning. Basically, the idea is that instead of defining a reward function, we have to tell the system what the optimal action is at each time step. The difficulty will thus be to reliably and consistently label the training data with turn-taking actions. This might be possible for actions such as TAKE_TURN or YIELD_TURN, which mostly depend on turn boundaries. It is, however, known to be much more difficult for backchannels (see section 3.3.3). Once we define a reasonable (if not ideal) policy to label the turn-taking action to take at each step (e.g. involving voting among several annotators), we will learn the policy based on the features defined in section 5.2.1.

5.3 A Model of Turn-Taking for Spoken Language Generation

NB: This part of the work will be simplified if more time is required for the other sections.

5.3.1 Spoken Language Generation for Efficient Interaction

To achieve natural and efficient interaction, improving the system's reactive turn-taking behavior is not enough. It is also necessary to give the right turn-taking signals to the user so that they can anticipate the end of system's turns and provide useful, and timely, backchannel feedback. Currently, dialogue system designers put a heavy emphasis on the intelligibility of system prompts, often resorting to over-explicit full grammatical sentences, a speaking rate slower than average, and a neutral (or flat) prosody. It is also considered wise to use prompts that inhibit backchannels, since systems cannot deal with them appropriately. In addition, the vast majority of commercial systems use recorded speech, which sacrifices flexibility to intelligibility.

In contrast, our goal is to provide efficient and flexible turn-taking. Prompt design should therefore adhere to these principles as well. We will take a data-driven approach to prompt design. First the Let's Go Public data used in learning turn-taking behavior, will be analyzed in terms of lexical, syntactic, and prosodic choices to provide insight on the issues with current prompts and potential improvement. This data, however, will be tainted by the behavior of the current system. Therefore, we will also use a different data set with more natural interaction, possibly human-human. One option is to resort to a currently available corpus such as the HCRC MapTask corpus. The recording quality of these dialogues should be high enough so that synthesis models can be built out of it. The conversations should be task-oriented, yet as natural as possible (i.e. avoiding scripted dialogues). If we do not find a corpus that match our needs, we will conduct a collection experiment (possibly in a Wizard-of-Oz setting).

5.3.2 Improving System Prompt Wording

The data will be first analyzed in terms of syntax. The goal will be to answer questions such as ”what types of constituents can be used as a full utterance and under what circumstances?”. Litterature on human-human conversation [Schegloff, 1996] and dialogue system prompt design [Hansen et al., 1996, Pitt and Edwards, 2003, Harris, 2005]. Ultimately, we will write a set of guidelines for turn-taking oriented system prompt design. These guidelines will be applied to write or modify the system prompts in Let's Go Public and the second evaluation system.

5.3.3 Prosodic Models for Conversational Speech Synthesis

Based on the human-human data, we will build a prosodic model for conversational speech synthesis. This model should capture turn-taking specific phenomena such as interruptions and fillers. Data-driven approaches to prosodic modeling will be used such as CART trees for duration. We will also rely on F0 unit selection, an algorithm designed by Raux and Black [2003]. We have shown that F0 unit selectoin produces significantly better F0 contours than rule-based approaches, while being much faster to build. These prosodic models will be built in a task-independent fashion, the idea being that they could be applied to any general or system-specific voice.

These models will be first evaluated through listening tests before being integrated in our evaluation dialogue systems. For example, human raters could be asked to listen to dialogues where one (or both) of the participants are synthesized and asked to grade them.

5.4 Summary and Work Items

The goals of this part of the work are to design models of turn-taking behavior both for perception/reaction and for speech generation. The work items involved and their current status are described in the following table.

Proposed	Design a model of reactive turn-taking behavior
Proposed	Implement the model in Apollo
Proposed	Write system prompt design guidelines
Proposed (if time permits)	Design a model of conversational prosody taking into account turn-taking signals
Completed	Design and implement an algorithm for flexible F0 generation

Chapter 6

Evaluation of Turn-Taking in Spoken Dialogue Systems

6.1 Systems for Turn-Taking Evaluation

6.1.1 Existing Application: The Let's Go Bus Information System

Base System

The Let's Go bus information system, already briefly described in section 2.2.1 is a telephone-based system that gives bus schedule information for the Pittsburgh area. Although it was first designed for in-lab experiments, it has been opened to the general public in March 2005, through the Port Authority Customer Service phone line. When Port Authority users call Customer Service outside the hours when human operators are available (which are weekdays from 7am to 7pm and weekends and holidays from 8am to 6pm), they are given the possibility to use the Let's Go system to get information.

The use of a publicly available system to evaluate our research presents many advantages. The system provides a constant flow of data that can be used to train and test models. Alternative versions of the system can be run in parallel (the system choosing the version at random for each call), allowing for fast, informed decisions on virtually any aspect of the system (one week brings approximately 300 usable dialogues). Last but not least, it ensures the research is grounded in the real world and allows to quantify the impact of each research idea on actual systems. On the other hand, a public system also has constraints that are not all related to the research issues under investigation. Since actual users rely on the system to get their bus schedule information, we must guarantee the best service possible. Therefore, to avoid confusion and make sure no erroneous information is used by the system, we originally restricted the interaction to system-directed dialogue and systematically use explicit confirmation. These somewhat conservative decisions were made with the idea that, as we would gain experience and learn more about our user population, we could relax the constraints towards more flexible dialogues without harming the usability of the system.

Getting Useful Data for Evaluation

A central issue with practical systems is the trade off between naturalness of interaction and dialogue effectiveness. While, ideally, we would like users to interact with our systems just as they converse with humans, in practice, many factors alter their behavior. In particular, user's turn-taking behavior might be very different from that observed in human-human dialogues.

First, the quality of speech understanding is still far from human performance, particularly in difficult conditions such as calls from noisy environments. As a result, the sole purpose of many dialogue turns is to avoid or correct understanding errors. In Raux et al. [2005], we describe the error recovery strategies used in Let's Go. They include asking the user to repeat, giving different levels of help, and backing off to neighborhood information when the system fails to recognize the exact departure stop.

The exact wording of system prompts also significantly affects user behavior. To investigate this, we wrote three different opening prompts for Let's Go:

- **Closed prompt**: ”Which bus number or departure place do you want information for?”
- **Semi-open prompt**: ”What bus information are you looking for?”
- **Open prompt**: ”What can I do for you?”

During one month (August 2005), we set the public system to pick one of those prompts at random at the beginning of each dialogue. The analysis of this data shows that the user response is longest for the Open prompt (3.8 words), shortest for the Closed prompt (1.6 words), and in-between for the Semi-open prompt (2.3 words)[1].

[1] These results are based on the speech recognition results and might not therefore be the exact number of words the user spoke. However, the relative differences between conditions are still relevant.

This experiment shows one way of influencing user response, which can be used to elicit specific behaviors for our experiments, while avoiding the use of recruited subjects and pre-written tasks.

Still, due to the variety of users and calling conditions (e.g. cell phone vs land line, indoor vs outdoor), the quality of the recognition in a public system is even worse than what could be obtained in laboratory experiments. One solution to this is to improve the system's speech recognition module. We are currently using transcribed data to retrain the acoustic and language models used by our speech recognizers. The inherent variability of the data makes this a non-trivial problem. For example, excluding some of the data (e.g. the noisiest utterances) might provide better performance than using the whole set. The problem is to determine which utterances are appropriate for training and which are not. One way to compensate for channel variability is channel normalization (e.g. CDCN, see Acero [1990]). Another potential source of improvement is to use information from the dialogue to constrain the language model. Currently, the system uses different language models at different dialogue states. We can further reduce perplexity by having different models for different bus routes or neighborhoods, which we can use when the user has given and confirmed these pieces of information. This should help the recognition of stop names, a crucial problem in Let's Go. Although improving speech recognition is not the goal of the proposed research, we will try and evaluate existing methods to maximize the accuracy and relevance of turn-taking evaluation.

If we are still not satisfied with the overall word error rate of the speech recognizer, we can exclude the worst dialogues from the data and concentrate on the better ones. In the data transcribed so far, we found that, although the overall word error rate is 56%, the average word error rate on the better half of the corpus is 30%.

Finally, in addition to Let's Go, other existing Olympus-based systems, such as RoomLine [Bohus, 2003] and TeamTalk [Harris et al., 2005], can be used to evaluate the proposed work with only minor modifications.

6.1.2 Applications Relevant for Turn-Taking Research

Task Features Relevant to Turn-Taking

Let's Go is a good example of current information access spoken dialogue systems. As such, it provides a good platform to investigate turn-taking issues and the potential impact of the proposed approach on state-of-the-art systems. However, in addition to improving the interaction for information access tasks, endowing systems with flexible turn-taking capabilities makes it possible to deal with new tasks, where the quality of the interaction is prominent. The main features of such new tasks that we wish to investigate are:

- **Bidirectional information flow**
  In typical information access tasks, the system has information and the user tries to obtain it. This results in dialogues that can be efficiently performed in two phases. First, the user communicates their information need to the system, potentially in a purely system-directed fashion. Second, the system provides the required information and, if necessary, the user can refine or modify their request. Even in mixed-initiative systems, which allow the user to introduce new information at any time, the system almost always has the initiative (i.e. it is in charge of moving the dialogue forward) in the first phase and the user has it in the second phase. Turn-taking in such systems is constrained by this fairly fixed, albeit implicit, initiative model. In general, turns are strictly organized in question-answer pairs and turn transitions are easily detectable, either because they correspond to complete questions or to a complete answer to the previous question.

  Bidirectional information flow happens when both participants have information that the other does not. In such cases, initiative is owned by the participant that has the relevant information at each dialogue state. As a result, initiative shifts (e.g. when a participant replies to a question by a question) are more likely to happen than in monodirectional information access tasks. Proper handling of bidirectional conversations requires the use of more advanced turn-taking models, since the structure of the dialogue is less predictable. The tasks of the TRAINS [Ferguson et al., 1996] and TRIPS [Ferguson and Allen, 1998, Allen et al., 2005] projects are examples of bidirectional information flow.

- **Complex information content**
  Most traditional information access systems provide an interface to a relational database. Both the information need and the information can be expressed as sets of atomic values such as numbers (e.g. times, room numbers, bus numbers...) or strings from a fixed set (movie titles, bus stop names...). Therefore, although long utterances combining several pieces of information at a time are sometimes allowed, most of the turns are very short, providing little opportunity for rich turn-taking phenomena such as backchannels and interruptions.

  Many conversations between humans are not reducible to exchanges of predefined information atoms. For example, when a student explains how they obtained a given result to a teacher, or when two people engage in social talk. This type of content yields longer turns, whose boundaries are harder to predict and which leave room for backchannels and interruptions. Although building systems for such tasks presents many challenges besides turn-taking, some existing systems tend towards more complex conversational content. This is the case of tutoring dialogue systems such as AutoTutor [Graesser et al., 2004] or ITSpoke [Litman and Silliman, 2004].

- **Interaction with real-time events**
  Most dialogue systems address domains for which it is reasonable to assume that the world does not change during the course of a dialogue. This reduces the need for interruptions and other real-time turn-taking actions. In domains such as teleoperating of mobile robots or car navigation, the system receives task-relevant information from the world during the dialogue. Being able to know whether, when, and how to provide information, potentially interrupting an on-going turn, is then an important feature of the dialogue system. Examples of such systems are WITAS [Lemon et al., 2001], a system to control an unmanned helicopter, and CMU's TeamTalk [Harris et al., 2005].

Proposed Applications for Turn-Taking Evaluation

A system for turn-taking evaluation should address a domain that has at least some of the features described above. On the other hand, aspects of the system such as dialogue and task management should be kept as simple as possible so as to focus the research on turn-taking. This balance is not easy to realize because rich turn-taking phenomena are often associated with complex and/or unpredictable content. At this point, we consider two alternative tasks for the evaluation system.

First, in the tutoring domain, we have started to work with a team including researchers in tutoring dialogue systems and psychologists towards a ”simulated tutee”. This system would play the role of a peer student, to which the user (a human student) will have to explain what they have learned in a lesson previously taught in class. Tutoring will be based on given problems that the (automated) tutee has to solve with the help of the (human) tutor. Tutoring features the first two characteristics described in the previous section. Information flow is more symmetric than in information access systems because both the tutor and the tutee have to explain their reasoning. This is all the more true in the peer learning situation where the two participants are at a closer level than in the teacher/student situation. Also, the information content of both the system and user turns will be more complex since they involve explaining one's path of thought towards a certain solution. To build this system, we will leverage the research done in the HCII on the Geometry Tutor and reuse both the set of problems and the knowledge representation from the current tutor. With the help of a psychologist specialized in peer learning, we will design dialogues that reflect a plausible behavior for a human tutee. Learning on the tutee's side will in fact be simulated because the system knows in advance what the tutor is supposed to teach it. This will help keeping the dialogue on track even in the presence of speech recognition and understanding errors.

An alternative domain is the ”Roboceptionist” task [Gockley et al., 2005], where an embodied artificial agent is in charge of welcoming visitors to a building and providing simple guidance such as where to find a specific room or person. The focus of this project is social interaction between humans and artificial agents. The first roboceptionist, Valerie, was deployed in CMU's Newell-Simon hall in November 2003 and has been recently replaced by Tank, a new character with similar features (but a different background story). These agents, in addition to providing directory information and directions, are endowed with a specific personality and background history. In the course of the interaction with human users, they provide information about themselves, their interests, what happened to them recently... Although currently the input modality is typed text through a keyboard placed in front of the agent's head (a graphical face drawn on a rotating flat panel display), we will replace this method by speech input. Since there are many issues in using speech in a crowded hall (where the roboceptionist works), we will first work on a duplicate version of the roboceptionist in a quieter, more controlled environment. The roboceptionist provides a domain where complex information is exchanged, because of the social content of the discussion. In addition, since the focus is on social interaction, the social function of turn-taking signals, backchannels, and interruptions make them particularly important to this task.

Finally, in addition to these two possible new systems, the TeamTalk system [Harris et al., 2005], already built on the Olympus architecture, provides another possible evaluation platform. In this project, a human operator interacts through spoken dialogue with a team of robots in a treasure hunt scenario. This domain features the third characteristic that make turn-taking an important feature of the conversation: the interaction with real-time events. For example, one of the robot might encounter an obstacle or find a certain person and want to inform the user. When is it relevant to give the user this information? What if the robot is already speaking at that time? What if the user is speaking to the robot? What if the user is speaking to another robot? The answer to these questions require taking into account both the task (e.g. how important the event is with regards to the current goal?), and turn-taking conventions (e.g. when is a good time to interrupt a conversation between two agents?).

6.2 An Evaluation Framework for Turn-Taking

Collecting data through dialogue systems is only part of the evaluation process. To gauge the absolute quality of the systems and compare the relative performance of different algorithms, we need metrics that quantify the quality of a dialogue's turn-taking. Unfortunately, there are currently no such metrics in the dialogue systems community. This part of the proposed work aims at designing and studying metrics for turn-taking.

6.2.1 Global Metrics

Global, or holistic, metrics, evaluate each dialogue as a whole. User satisfaction is arguably the ultimate standard by which dialogue systems should be judged. It is obtained through questionnaires filled in by subjects after they have interacted with the system(s). The problem is that it is a very subjective metric and that different people might use the grading scale in different ways. For these reasons, user satisfaction, while often collected, does not always give a good insight on the differences between systems (except when these differences are very large). Task completion rate is more often used, particularly in settings where users are given specific goals to achieve through their interaction with the system. In this case, since the goals are clear, task completion rate is a fairly objective metric. For the Let's Go system, because we do not know in advance what the goal of the user is, we found that it is not always trivial to determine task completion. Dialogues have to be judged as successful or not by human raters, given explicit criteria. In a previous study [Raux et al., 2005], we found a kappa of 0.75 between two human raters for task success. Agreement can be improved by using part of the data to train the raters and to let them negotiate the success criteria. For a tutoring system, task completion cannot be defined in the traditional way. Rather, learning gains for the user are measured using pre- and post-tests. Similarly, specific metrics have to be defined to evaluate the quality of social dialogue (e.g. how often to previous users reuse the system? how long do they interact with it?). Dialogue length (in number of turns or time spent) is also often used to evaluate task-oriented systems. Considering that the proposed approach aims at reducing the time spent for each turn, dialogue duration (in seconds) seems the appropriate measure. However, for tutoring systems, more time spent on a topic might actually mean better learning so the total duration of the dialogue is not a good metric. An alternative metric, specifically targeting turn-taking, is average turn duration (dialogue duration divided by total number of turns).

6.2.2 Local Metrics

While global metrics are good to quantify the impact of a change to the system on the overall performance, they are often too coarse to understand the specific strengths and weaknesses of the tested approach. Local metrics, which measure the quality of the dialogue at every point in time (or at each turn), can better inform future research and improvements. The theory of Sacks et al [Sacks et al., 1974] offers one possible metric for turn-taking. They hypothesize that humans tune their turn-taking behavior so as to *minimize the duration of gaps and overlaps*. Following this idea, objective metrics of turn-taking at the turn level can be defined as the duration of the gaps (i.e. time when neither participant is speaking) before and after the turn and total duration of overlapping speech (i.e. time when the two participants speak at the same time) during this turn. These metrics alone shouldn't drive turn-taking behavior though. For example, backchannels typically result in overlaps but are not considered to negatively affect the dialogue (rather the opposite). The amount of information exchanged either at the task or grounding (e.g. backchannels) level should also be taken into account, as a positive measure of the quality of a turn. Another approach is to have human annotators grade turn-taking actions (most likely in a binary way: appropriate/inappropriate). This is the approach that we have taken in the preliminary experiment described in Chapter 2. We will refine the proposed labelling scheme and build a labeling tool adapted to rating turn-taking actions. Several annotators will be trained and inter-annotator agreement computed to validate the final labeling scheme.

6.2.3 Relationship Between Local and Global Metrics

Ultimately, we want to get the best of both global and local metrics. That is, we want to know when specific turn-taking problems tend to occur and which ones, but also how much they impact the overall performance of the system. In order to study the relationship between the various global and local metrics proposed, we will perform a regression analysis similar to the PARADISE spoken dialogue evaluation framework [Walker et al., 1997] or the work of Bohus and Rudnicky on misunderstandings and non-understandings [Bohus and Rudnicky, 2005c]. Besides its main goal of furthering our understanding of turn-taking behavior in spoken dialogue systems, this study will aim at defining automatically computable metrics that correlate well with metrics that require human labelling. If a metric with satisfactory correlation can be defined, this would eliminate (or at least reduce) the need for hand-labelling data, a costly operation, in future experiments.

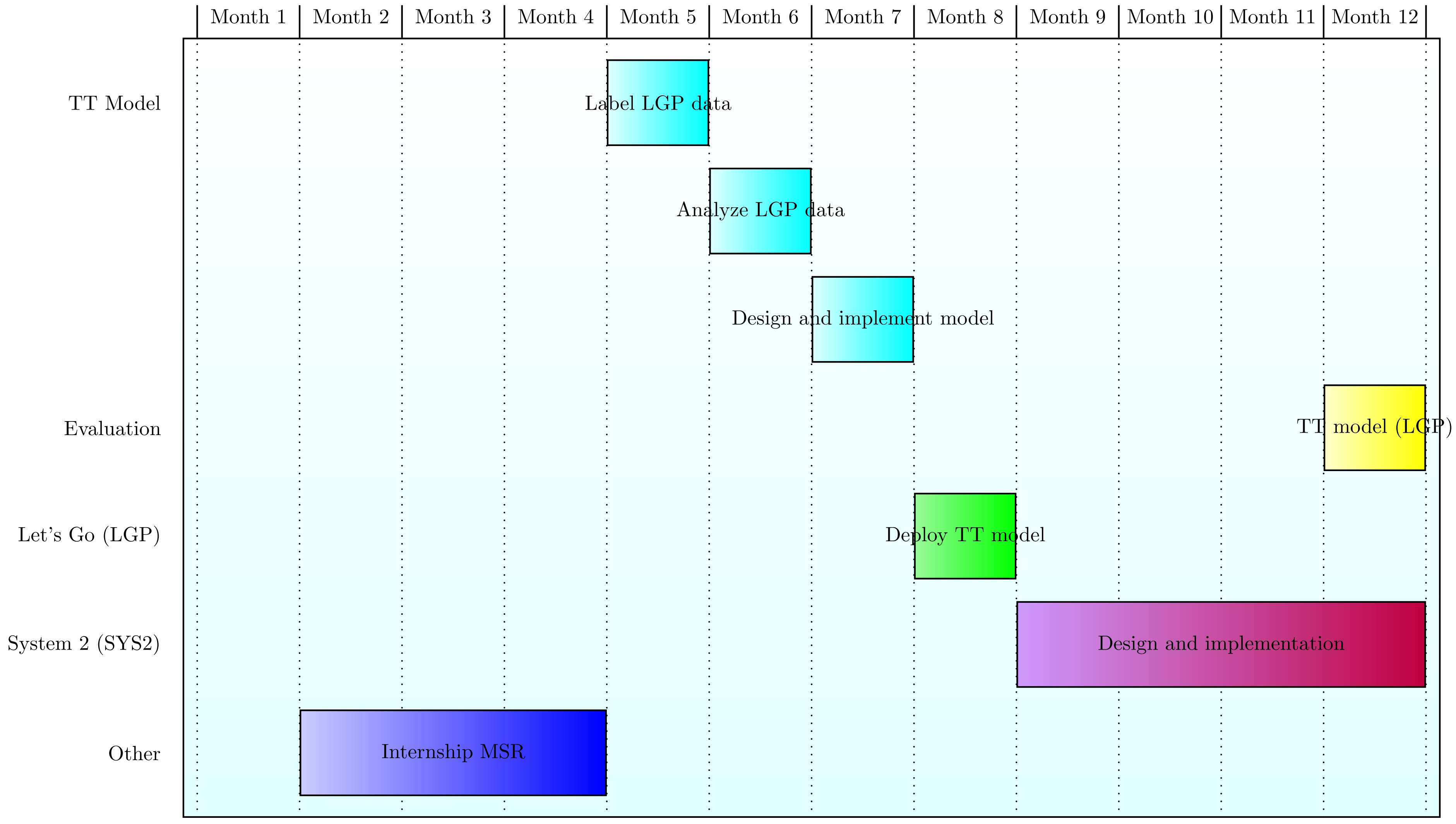
6.3 Summary and Work Items

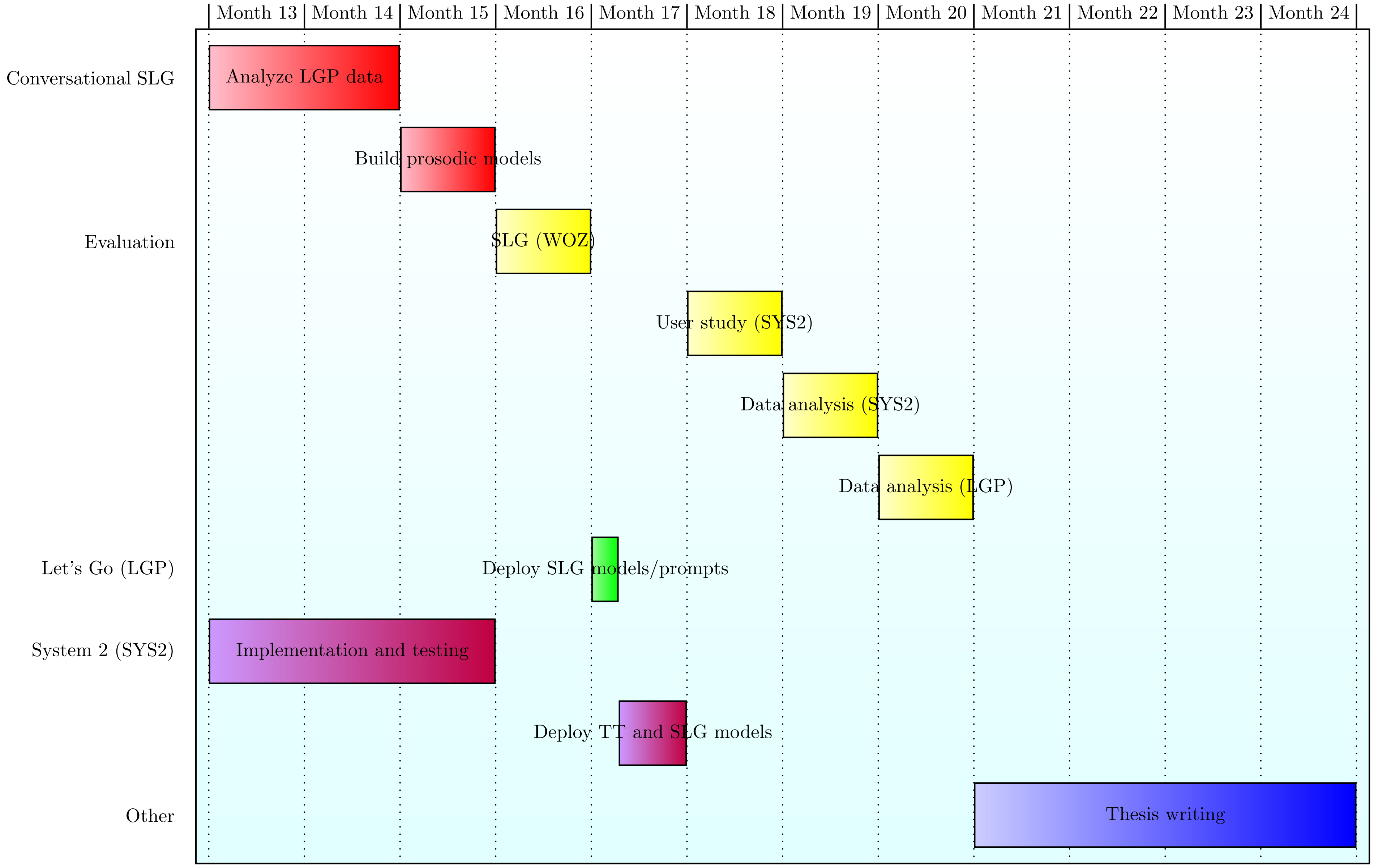
The goals of this part of the work are to define an evaluation framework for spoken dialogue systems that takes into account interaction quality and to evaluate the models and algorithms proposed in the previous section on practical applications. The work items involved and their current status are described in the following table.

Completed	Implement Let's Go
Proposed	Implement a second system
Proposed	Define and assess a set of global metrics of dialogue quality
Proposed	Define and assess a set of local metrics of interaction quality
Proposed	Define and validate an integrated evaluation framework
Proposed	Evaluate the reactive model of turn-taking
Proposed (if time permits)	Evaluate the lexical/syntactic/prosodic model of conversational SLG
Proposed	Evaluate all the proposed architecture and models together

Appendix A

Proposed Thesis Timeline


Month 1
Month 2
Month 3
Month 4
Month 5
Month 6
Month 7
Month 8
Month 9
Month 10
Month 11
Month 12
TT Model
Label LGP data
Analyze LGP data
Design and implement model
Evaluation
TT model (LGP)
Let’s Go (LGP)
Deploy TT model
System 2 (SYS2)
Design and implementation
Other
Internship MSR


Month 13
Month 14
Month 15
Month 16
Month 17
Month 18
Month 19
Month 20
Month 21
Month 22
Month 23
Month 24
Conversational SLG
Analyze LGP data
Build prosodic models
Evaluation
SLG (WOZ)
User study (SYS2)
Data analysis (SYS2)
Data analysis (LGP)
Let's Go (LGP)
Deploy SLG models/prompts
System 2 (SYS2)
Implementation and testing
Deploy TT and SLG models
Other
Thesis writing

Bibliography

A. Acero. *Acoustical and Environmental Robustness in Automatic Speech Recognition*. PhD thesis, Carnegie Mellon University, 1990. 6.1.1

G. S. Aist. Expanding a time-sensitive conversational architecture for turn-taking to handle content-driven interruption. In *Proc. ICSLP 1998*, Sydney, Australia, 1998. 3.2.2

G. S. Aist and J. Mostow. Adapting human tutorial interventions for a reading tutor that listens: Using continuous speech recognition in interactive educational multimedia. In *Proc. CALL '97 Conference on Multimedia*, Exeter, England, 1997. 3.2.2

G. S. Aist and J. Mostow. Measuring the effects of backchanneling in computerized oral reading tutoring. In *Proc. ESCA Workshop on Prosody and Dialogue*, Eindhoven, Netherlands, 1999. 3.2.2

J. F. Allen, D. Byron, M. Dzikovska, G. Ferguson, L. Galescu, and A. Stent. Towards a generic dialogue shell. *Natural Language Engineering*, 6(3):1–16, 2000. 3.2.1

J. F. Allen, G. Ferguson, and A. Stent. An architecture for more realistic conversational systems. In *Proc. Intelligent User Interfaces 2001 (IUI-2001)*, pages 1–8, Santa Fe, NM, 2001. 3.2.1, 5.2.1

J. F. Allen, G. Ferguson, A. Stent, S. Stoness, M. Swift, L. Galescu, N. Chambers, E. Campana, and G. S. Aist. Two diverse systems built using generic components fo spoken dialogue (recent progress on trips). In *Interactive Demonstration Track, Association of Computational Linguistics Annual Meeting*, Ann Arbor, MI, 2005. 3.2.1, 6.1.2

G. T. M. Altmann and M. J. Steedman. Interaction with context during human sentence processing. *Cognition*, 30(3):191–238, 1988. 3.2.1

V. Andronache and M. Scheutz. Contention scheduling: A viable action-selection mechanism for robotics? In *Proc. Thirteenth Midwest AI and Cognitive Science Conference*, pages 122–129. AAAI Press, 2002. 5.2.1

ACL Workshop on Incremental Parsing, 2004. *Proc. ACL Workhop on Incremental Parsing: Bringing Engineering and Cognition Together*, Barcelona, Spain, 2004. Association for Computational Linguistics. 3.2.1

G. W. Beattie. Turn-taking and interruption in political interviews: Margaret Thatcher and Jim Callaghan compared and contrasted. *Semiotica*, 39(1-2):93–114, 1982. 5.1

G. W. Beattie. *Talk: An Analysis of Speech and Non-Verbal Behaviour in Conversation*. Open University Press, 1983. 3.3.1

L. Bell, J. Boye, and J. Gustafson. Real-time handling of fragmented utterances. In *Adaptation in Dialogue Systems Workshop, NAACL*, Pittsburgh, Pennsylvania, 2001. 3.3.2

A. Black and K. Lenzo. Limited domain synthesis. In *ICSLP2000*, volume II, pages 411–414, Beijing, China., 2000. 4.1.1

B. Blumberg. Action-selection in hamsterdam: Lessons from ethology. In *From Animals to Animats, Proc. Third International Conference in the Simulation of Adaptive Behavior*, Brighton, UK, 1994. 5.2.1

D. Bohus. The roomline spoken dialog system. http://www.cs.cmu.edu/ dbohus/RoomLine, 2003. 6.1.1

D. Bohus and A. Rudnicky. Larri: A language-based maintenance and repair assistant. In *Proc. ISCA Tutorial and Research Workshop on Multi-Modal Dialogue in Mobile Environments*, Kloster Irsee, Germany, 2002. 1

D. Bohus and A. Rudnicky. RavenClaw: Dialog management using hierarchical task decomposition and an expectation agenda. In *Eurospeech03*, Geneva, Switzerland, 2003. 4.1.1, 4.1.2, 4.1.4

D. Bohus and A. Rudnicky. Sorry i didn't catch that... In *Proc. SIGdial 2005*, Lisbon, Portugal, 2005a. 4.1.1

D. Bohus and A. Rudnicky. Error handling in the ravenclaw dialog management architecture. In *Proc. HLT/EMNLP 2005*, Vancouver, BC, 2005b. 1, 4.1.1

D. Bohus and A. Rudnicky. A principled approach for rejection threshold optimization in spoken dialog systems. In *Proc. Interspeech 2005*, Lisbon, Portugal, 2005c. 6.2.3

J. Bos and T. Oka. An inference-based approach to dialogue system design. In *COLING 2002*, Taipei, Taiwan, 2002. 3.1

S. Brennan. Processes that shape conversation and their implications for computational linguistics. In *Proc. 38th Annual Meeting of the Association for Computational Linguistics*, Hong Kong, 2000. 1

R. A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14–23, 1985. 3.2.2

M. Bull. *The Timing and Coordination of Turn-Taking*. PhD thesis, University of Edinburgh, 1997. 2.4.4

M. Bull and M. Aylett. An analysis of the timing of turn-taking in a corpus of goal-oriented dialogue. In *ISCLP 98*, pages 1175–1178, Sydney, Australia, 1998. 2.4.4

J. Carletta, S. Isard, G. Doherty-Sneddon, A. Isard, J. C. Kowtko, and A. H. Anderson. The reliability of a dialogue structure coding scheme. *Computational Linguistics*, 23 (1):13–31, March 1997. 2.4.4

N. Cathcart, J. Carletta, and E. Klein. A shallow model of backchannel continuers in spoken dialogue. In *Proc. 10th Conference of the European Chapter of the Association for Computational Linguistics (EACL10)*, pages 51–58, Budapest, Hungary, 2003. 3.3.3

W. L. Chafe. *Talking Data: Transcription and Coding Methods for Language Research*, chapter Prosodic and Functional Units of Language, pages 33–43. Lawrence Erlbaum, 1992. 3.3.1

A. Cheyer and D. Martin. The open agent architecture. *Journal of Autonomous Agents and Multi-Agent Systems*, 4(1):143–148, March 2001. 3.1

P. M. Clancy, S. A. Thompson, R. Suzuki, and H. Tao. The conversational use of reactive tokens in english, japanese, and mandarin. *Journal of Pragmatics*, 26:355–387, 1996. 3.3.3

C. Darves and S. Oviatt. Adaptation of users' spoken dialogue patterns in a conversational interface. In *ICSLP 2002*, 2002. 2.5

M. Denecke and A. Waibel. Dialogue strategies guiding users to their communicative goals. In *Proc. Eurospeech '97*, Rhodes, Greece, 1997. 1

R. Denny. *Interaction Structure and Strategy*, chapter Pragmatically Marked and Unmarked Forms of Speaking-Turn Exchange, pages 135–174. Cambridge University Press, 1985. 3.3.3

K. Dohsaka and A. Shimazu. System architecture for spoken utterance production in collaborative dialogue. In *Working Notes of IJCAI 1997 Workshop on Collaboration, Cooperation and Conflict in Spoken Dialogue Systems*, Nagoya, Japan, 1997. 3.2.1

S. Duncan. Some signals and rules for taking speaking turns in conversations. *Journal of Personality and Social Psychology*, 23(2):283–292, 1972. 3.3.1

S. Duncan and D. W. Fiske. *Interaction Structure and Strategy*, chapter The Turn System, pages 43–64. Cambridge University Press, 1985. 3.3.3

J. Edlund, G. Skantze, and R. Carlson. Higgins - a spoken dialogue system for investigating error handling techniques. In *Proc. ICSLP*, Jeju, Korea, 2004. 1

F. Farfán, H. Cuayáhuitl, and A. Portilla. Evaluating dialogue strategies in a spoken dialogue system for email. In *Proc. IASTED Artificial Intelligence and Applications*, Manalmádena, Spain, 2003. 1

G. Ferguson and J. F. Allen. Trips: An integrated intelligent problem-solving assistant. In *Proc. Fifteenth National Conference on AI (AAAI-98)*, pages 26–30, 1998. 3.2.1, 6.1.2

G. Ferguson, J. F. Allen, and B. Miller. Trains-95: Towards a mixed-initiative planning assistant. In *Proc. Third Conference on Artificial Intelligence Planning Systems (AIPS-96)*, pages 70–77, Edinburgh, Scotland, 1996. 6.1.2

L. Ferrer, E. Shriberg, and A. Stolcke. Is the speaker done yet? Faster and more accurate end-of-utterance detection using prosody in human-computer dialog. In *ICSLP*, Denver, Colorado, 2002. 3.3.2

L. Ferrer, E. Shriberg, and A. Stolcke. A prosody-based approach to end-of-utterance detection that does not require speech recognition. In *ICASSP*, Hong Kong, 2003. 3.3.2

A. Ferrieux and M. D. Sadek. An efficient data-driven model for cooperative spoken dialogue. In *ICSLP 1994*, Yokohama, Japan, 1994. 1

C. E. Ford and S. A. Thompson. *Interaction and Grammar*, chapter Interactional Units in Conversation: Syntactic, Intonational, and Pragmatic Resources for the Management of Turns, pages 134–184. Cambridge University Press, 1996. 3.3.1, 3.3.1

H. Furo. *Turn-Taking in English and Japanese. Projectability in Grammar, Intonation, and Semantics.* Routeledge, 2001. 3.3.1, 3.3.1, 3.3.3

R. Gockley, A. Bruce, J. Forlizzi, M. Michalowski, A. Mundell, S. Rosenthal, B. P. Sellner, R. Simmons, K. Snipes, A. Schultz, and J. Wang. Designing robots for long-term social interaction. In *Proc. IROS 2005*, Edmonton, Canada, August 2005. 6.1.2

C. Goodwin. *Conversational Organization: Interaction between Speakers and Hearers*. Academic Press, 1981. 3.3.1

A. C. Graesser, S. Lu, G. T. Jackson, H. Mitchell, M. Ventura, A. Olney, and M. M. Louwerse. Autotutor: A tutor with dialogue in natural language. *Behavioral Research Methods, Instruments, and Computers*, 36:180–193, 2004. 6.1.2

B. Hansen, D. Novick, and S. Sutton. Systematic design of system prompts. In *Prof. Conference on Human Factors in Computing Systems (CHI'96)*, pages 157–164, Vancouver, BC, 1996. 5.3.2

R. Hariharan, J. Häkinen, and K. Laurila. Robust end-of-utterance detection for real-time speech recognition applications. In *ICASSP*, Salt Lake City, May 2001. 3.3.2

R. A. Harris. *Voice Interaction Design*. Morgan Kaufmann, Cambridge, Mass., 2005. 5.3.2

T. K. Harris, S. Banerjee, and A. Rudnicky. Heterogeneous multi-robot dialogues for search tasks. In *Proc. AAAI Workshop on Dialogical Robots: Verbal Interaction with Embodied Agents and Situated Devices*, Stanford, CA, 2005. 4.1.1, 6.1.1, 6.1.2, 6.1.2

H. Hassan, A. Crespo, and J. Simó. Flexible real-time architecture for hybrid mobile robotic applications. In *Proc. 9th International Federation of Automatic Control Symposium*, Budapest, Hungary, 2000. 3.2.2

X. Huang, F. Alleva, H.-W. Hon, K.-F. Hwang, M.-Y. Lee, and R. Rosenfeld. The SPHINX-II speech recognition system: an overview. *Computer Speech and Language*, 7(2):137–148, 1992. 4.1.1, 4.1.2

M. Humphrys. Action selection methods using reinforcement learning. In *From Animals to Animats 4: Proc. Fourth International Conference on Simulation of Adaptive Behavior (SAB-96)*, pages 135–144, Cape Cod, MA, 1996. 5.2.1

J. Jaffe and S. Feldstein. *Rhythms of Dialogue*. Academic Press, 1970. 2.4.1

Y. Kamide, G. T. M. Altmann, and S. L. Haywood. The time-course of prediction in incremental sentence processing: Evidence from anticipatory eye movements. *Journal of Memory and Language*, 49:133–156, 2003. 3.2.1

Y. Kato, S. Matsubara, Toyama K., and Y. Inagaki. Incremental dependency parsing based on headed context free grammar. *Systems and Computers in Japan*, 36(2):84–97, 2005. 3.2.1

A. Kendon. Some functions of gaze direction in social interaction. In *Acta Psychologica*, volume 26, 1967. 3.3.1

H. Koiso, Y. Horiuchi, S. Tutiya, A. Ichikawa, and Y. Den. An analysis of turn-taking and backchannels based on prosodic and syntactic features in japanese map task dialogs. *Language and Speech*, 41(3-4):295–321, 1998. 3.3.1, 3.3.3

O. Lemon. Context-sensitive speech recognition in isu dialogue systems: results for the grammar switching approach. In *Proc. CATALOG, 8th Workshop on the Semantics and Pragmatics of Dialogue*, Barcelona, Spain, 2004. 1

O. Lemon, A. Bracy, A. Gruenstein, and S. Peters. The witas multi-modal dialogue system i. In *Proc. Eurospeech 2001*, Aalborg, Danemark, 2001. 1, 6.1.2

O. Lemon, L. Cavedon, and B. Kelly. Managing dialogue interaction: A multi-layered approach. In *Proc. SIGdial Workshop 2003*, Sapporo, Japan, 2003. 3.2.2

W. J. M. Levelt. *Speaking: from Intention to Articulation*. MIT Press, 1993. 3.3.1

D. J. Litman and S. Silliman. Itspoke: An intelligent tutoring spoken dialogue system. In *Proc. Human Language Technology Conference: 4th Meeting of the North American Chapter of the Association for Computational Linguistics (HLT/NAACL)*, Boston, MA, 2004. 6.1.2

P. Maes and R. A. Brooks. Learning to coordinate behaviors. In *Proc. AAAI*, pages 796–802, Boston, MA, 1990. 5.2.1

D. Mori, S. Matsubara, and Y. Inagaki. Incremental parsing for interactive natural language interface. In *Proc. IEEE International Conference on Systems, Man and Cybernetics*, Tucson, AZ, 2001. 3.2.1

J. P. Muller. *The Design of Intelligent Agents: A Layered Approach*. Springer, 1996. 3.2.2

M. Nakano, K. Dohsaka, N. Miyazaki, J. Hirasawa, M. Tamoto, M. Kawamori, A. Sugiyama, and T. Kawabata. Handling rich turn-taking in spoken dialogue systems. In *Eurospeech*, Budapest, Hungary, 1999a. 3.2.1

M. Nakano, N. Miyazaki, J. Hirasawa, K. Dohsaka, and T. Kawabata. Understanding unsegmented user utterances in real-time spoken dialogue systems. In *ACL*, College Park, MD, 1999b. 3.2.1

M. Nakano, N. Miyazaki, N. Yasuda, N. Sugiyama, J. Hirasawa, K. Dohsaka, and K. Aikawa. Wit: A toolkit for building robust and real-time spoken dialogue systems. In *1st SIGdial Workshop*, Hong-Kong, 2000. 3.2.1

Nuance. *OpenSpeech(TM) Dialog*. URL `http://www.nuance.com/dialog/`. 4.1.1

A. Oh and A. Rudnicky. Stochastic language generation for spoken dialogue systems. In *ANLP/NAACL 2000 Workshop on Coversational Systems*, pages 27–32, Seattle, WA, 2000. 4.1.1

B. Oreström. *Turn-Taking in English Conversation*. CWK Gleerup, Lund, 1983. 3.3.1

I. Pitt and A. Edwards. *Design of Speech-based Intrefaces*. Springer, London, UK, 2003. 5.3.2

R. Porzel and M. Baudis. The tao of chi: Towards effective human-computer interaction. In *HLT/NAACL 2004*, Boston, MA, 2004. 1, 3.1

A. Raux and A. Black. A unit selection approach to f0 modeling and its application to emphasis. In *ASRU2003*, St Thomas, Virgin Is., 2003. 5.3.3

A. Raux, B. Langner, A. Black, and M. Eskenazi. LET'S GO: Improving spoken dialog systems for the elderly and non-native. In *Eurospeech03*, Geneva, Switzerland, 2003. 4.1.1

A. Raux, B. Langner, D. Bohus, A. W. Black, and M. Eskenazi. Let's go public! taking a spoken dialog system to the real world. In *Proc. Interspeech 2005*, Lisbon, Portugal, 2005. 4.1.1, 6.1.1, 6.2.1

C. P. Rose, A. Roque, D. Bhembe, and K. VanLehn. An efficient incremental architecture for robust interpretation. In *Proc. Human Languages Technology Conference*, San Diego, CA, 2002. 3.2.1

A. Rudnicky and W. Xu. An agenda-based dialog management architecture for spoken language systems. In *IEEE Automatic Speech Recognition and Understanding Workshop*, Keystone, CO, 1999. 4.1.1

A. Rudnicky, E. Thayer, P. Constantinides, C. Tchou, R. Shern, K. Lenzo, Xu W., and A. Oh. Creating natural dialogs in the Carnegie Mellon Communicator system. In *Eurospeech99*, volume 4, pages 1531–1534, Budapest, Hungary, 1999. 3.1

A. Rudnicky, C. Bennett, A. Black, A. Chotimongkol, K. Lenzo, A. Oh, and R. Singh. Task and domain specific modelling in the Carnegie Mellon Communicator system. In *ICSLP2000*, volume II, pages 130–133, Beijing, China, 2000. 4.1.1

H. Sacks, E. A. Schegloff, and G. Jefferson. A simplest systematics for the organization of turn-taking for conversation. *Language*, 50(4):696–735, 1974. 1, 3.3.1, 5.2.1, 6.2.2

R. Sato, R. Higashinaka, M. Tamoto, M. Nakano, and K. Aikawa. Learning decision trees to determine turn-taking by spoken dialogue systems. In *ICSLP 2002*, Denver, CO, 2002. 3.2.1, 3.3.2

D. Schaffer. The role of intonation as a cue to turn taking in conversation. *Journal of Phonetics*, 11:243–257, 1983. 3.3.1

E. A. Schegloff. *Interaction and Grammar*, chapter Turn organization: one intersection of grammar and interaction, pages 52–133. Cambridge University Press, 1996. 5.3.2

S. Seneff, E. Hurley, R. Lau, C. Pao, P. Schmid, and V. Zue. Galaxy-ii: A reference architecture for conversational system development. In *ICSLP98*, Sydney, Australia., 1998. 3.1, 4.1.1

S. Seneff, R. Lau, and J. Polifroni. Organization, communication, and control in the galaxy-ii conversational system. In *Proc. Eurospeech-99*, Budapest, Hungary, 1999. 3.1

M.-L. Sorjonen. *Interaction and Grammar*, chapter On repeats and responses in Finnish conversations, pages 277–327. Cambridge University Press, 1996. 3.3.1

A. Stent, J. Dowding, J.M. Gawron, E. Owen Bratt, and R. Moore. The CommandTalk spoken dialogue system. In *Proc. ACL-99*, College Park, MD, 1999. 3.1

S. C. Stoness, J. Tetreault, and J. Allen. Incremental parsing with reference interaction. In *Proc. ACL Workshop on Incremental Parsing*, pages 18–25, Barcelona, Spain, 2004. 3.2.1

L. ten Bosch, N. Oostdijk, and J.P. de Ruiter. Durational aspects of turn-taking in spontaneous face-to-face and telephone dialogues. In *Conference on Text, Speech and Dialogue*, Brno, Czech Republic, September 2004. 2.4.1

K. R. Thorisson. *Multimodality in Language and Speech Systems*, chapter Natural Turn-Taking Needs No Manual: Computational Theory and Model, From Perception to Action, pages 173–207. Kluwer Academic Publishers, 2002. 3.2.2

K. R. Thorisson. *Communicative Humanoids: A Computational Model of Psychosocial Dialogue Skills*. PhD thesis, Massachusetts Institute of Technology, 1996. 3.2.2

K. R. Thorisson. A mind model for communicative humanoids. *International Journal of Applied Artificial Intelligence*, 13(4-5):449–486, 1999. 3.2.2

L. K. Tyler and W. D. Marlsen-Wilson. The on-line effects of semantic context on syntactic processing. *Journal of Verbal Learning and Verbal Behaviour*, 16:683–692, 1977. 3.2.1

M. A. Walker, D. J. Litman, C. A. Kamm, and A. Abella. Paradise: A framework for evaluating spoken dialogue agents. In *Proc. 35th Annual Meeting of the Association for Computational Linguistics (ACL-97)*, Madrid, Spain, 1997. 6.2.3

M. A. Walker, A. Rudnicky, R. Prasad, J. Aberdeen, E. Owen Bratt, J. Garofolo, H. Hastie, A. Le, B. Pellon, A. Potamianos, R. Passoneau, S. Roukos, G. Sanders, S. Seneff, and D. Stallard. Darpa communicator: Cross-system results for the 2001 evaluation. In *Proc. ICSLP 2002*, Denver, CO, 2002. 1

K. Wang. Semantic synchronous understanding for robust spoken language understanding applications. In *Proc. ASRU 2003*, St Thomas, US Virgin Islands, 2003. 3.2.1

N. Ward and W.. Tsukahara. Prosodic features which cue back-channel responses in english and japanese. *Journal of Pragmatics*, 32:1177–1207, 2000. 3.3.3

N. Ward, A. Rivera, K. Ward, and D. Novick. Root causes of lost time and user stress in a simple dialog system. In *Interspeech 2005*, Lisbon, Portugal, 2005. 1, 3.1

W. Ward and S. Issar. Recent improvements in the CMU spoken language understanding system. In *Proceedings of the ARPA Human Language Technology Workshop*, pages 213–216, 1994. 4.1.1, 4.1.2

W. Wesseling and R.J.J.H. van Son. Timing of experimentally elicited minimal responses as quantitative evidence for the use of intonation in projecting TRPs. In *Interspeech 2005*, pages 3389–3392, Lisbon, Portugal, 2005. 3.3.1

M. Wiren. *Studies in Incremental Natural Langauge Analysis*. PhD thesis, Linkoping University, 1992. 3.2.1