MODALITY TRACKING IN THE MULTIMODAL BELL LABS COMMUNICATOR

Alexandros Potamianos[0] *Egbert Ammicht*[1] *and Eric Fosler-Lussier*[2]

[0] Dept. of Elec. & Comp. Engineering, Technical Univ. of Crete, Chania 73100, Greece
[1] Bell Labs, Lucent Technologies, 600 Mountain Ave., Murray Hill, NJ 07974, U.S.A.
[2] Dept. of Computer and Information Science, Ohio State University, Columbus, OH 43210, U.S.A.
potam@telecom.tuc.gr eammicht@lucent.com fosler@cis.ohio-state.edu

ABSTRACT

In this paper, we describe our efforts in designing and building the multimodal Bell Labs Communicator system. Several innovations are necessary for generalizing the speech-only user interface and the semantic/pragmatic algorithms to handle both speech and visual modalities, including changes to the parser, semantic and pragmatic modules to better integrate the user input from different modalities within and across dialogue turns, as well as extending the concept of e-forms to include the visual modality. The second part of the paper describes the design of a natural, consistent and efficient multimodal user interface. We introduce the concept of the "dominant modality": the system adaptively tracks the most probable interaction mode and suggests that modality to the user. Modality tracking provides an elegant solution to the "turn-taking" issue in multimodal spoken dialogue systems.

1. INTRODUCTION

Mobile terminals present new interface design challenges that can be addressed by a combination of speech and visual modalities [2, 10]. Limited screen real estate, the absence of a keyboard, and the usage of terminals in unpredictable mobile environments, set new constraints. Thus, multimodal interfaces have emerged that support speech input and output, as well as visual input and output.

The main goal of the Bell Labs Communicator program is to design a domain-independent and modality-independent platform that can be used to quickly prototype state-of-the-art spoken dialogue systems for a large range of applications that include form-filling, database queries and navigation of query results [12, 1, 6]. In this work, we test the validity of these claims by extending the Communicator platform to include the visual modality. We show how the semantic representation, semantic/pragmatic modules and interaction/dialogue manager of the Communicator can be directly ported to a new modality. The main features of the Communicator platform that make porting easy include the hierarchical representation of the domain ontology, the semantic/pragmatic algorithms used for merging information from various sources, the efficient ambiguity representation and resolution mechanisms and the dynamic e-form representation in the dialogue manager.

A second important contribution of this paper is the design of the multimodal user interface: an interface that is consistent, efficient and balanced between the visual and speech modalities. We introduce a dominant modality tracking algorithm that predicts the modality of choice for each user and for each point in the human-machine interaction, and suggests it to the user. The proposed modality tracking algorithm merges the "open-mike" and "click-to-talk" interaction modes typically found in multimodal spoken dialogue systems. Adaptive modality tracking can significantly enhance the user experience compared to traditional multimodal interfaces by providing more freedom to the user and better adapting to the user expectation and needs.

2. COMMUNICATOR DIALOGUE SYSTEM

The Communicator system consists of four basic natural language modules and sub-modules: a) the semantic module that parses, interprets and extracts the attribute-value candidates for each user utterance, b) the pragmatic module that performs the context tracking, domain act classification, pragmatic analysis and pragmatic scoring c) the dialogue manager and initiative tracking module, and d) the natural language generation module. In this section, we will briefly describe these modules; details can be found in [1]. The description of the speech recognizer, text-to-speech system and i/o platform are also beyond the scope of this paper (see [12, 15] for details).

[†]Most of this work was done while Alexandros Potamianos and Eric Fosler-Lussier were with Bell Labs, Lucent Technologies, 600 Mountain Ave., Murray Hill, NJ 07974, U.S.A. This work was partially supported by DARPA under the auspices of the Communicator Project.

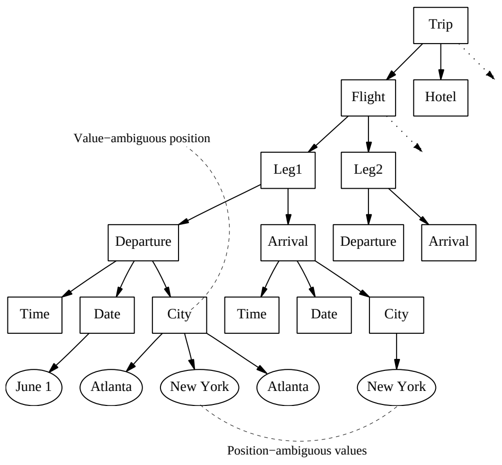


Fig. 1. Notepad tree illustrating value ambiguity (departure city Atlanta or New York) and position ambiguity (New York).

2.1. Semantic Module

Three related hierarchical data structures are used for representing and instantiating domain semantics. The **prototype tree** encodes the domain ontology as a set of *is-a* or *has-a* relationships (*e.g.,* a flight has one or more segments or legs). This provides the skeleton for the **notepad tree**, which holds all raw values elicited or inferred from user input. Data are defined in terms of the path from the tree root to a leaf (the attribute), and the associated value (also known as an *attribute-value (AV) pair*). AV pairs can be ambiguous due to natural language ambiguity, speech recognition errors or conflicting information in user input. We model two types of ambiguities in the system: *value ambiguities*, where the system is unsure of the value for a particular attribute, and *position ambiguities*, where the attribute corresponding to a particular value is ambiguous. Figure 1 illustrates both types of ambiguities.

Since raw data can be incomplete or inconsistent, the semantic module (described below) derives candidate AV pairs, which are held in a third structure, the **application tree**. Candidate selection is performed by scoring AV pairs based on supporting evidence for or against the candidate. This evidence is provided by raw data, by pragmatic analysis and by matching a hypothesis to the current context.

In order to fill the notepad tree, raw data (either from typed natural language input or spoken input) is parsed using a recursive finite-state **parser** [13], and then transformed by a set of application-dependent **interpreter** routines, yielding a canonical form the notepad can use. For example, in the travel domain, cities are transformed into airport codes, date strings (e.g., “Tuesday”) are converted to the relevant date, and phrases like “first class” are changed into corresponding fare codes.

Processed data are placed in the notepad tree, and new candidate values, if any, are derived and placed in the application tree. These trees are also updated by the context tracking and pragmatic scoring algorithms described below.

2.2. Pragmatic Module

The system maintains an expected context for every user response expressed as a path r from the root to some node of the prototype tree. Values extracted from a user utterance are associated with a partial attribute, i.e., a partial path l. The **context tracking** algorithm (see [1]) matches the context r to the partial attribute l to form a complete path a. For example, in the simplest case of exact match, given the context r “.trip.flight.leg1“ and a datum with partial attribute l “.departure.city”, the complete path “.trip.flight.leg1.departure.city” is derived.

Any communicative act made by the user is classified by the pragmatic module as one of a number of *domain acts* which can manipulate the notepad and application trees. The default action is a “fill request,” which attempts to create candidate values from the user input. The context tracking module can be directed via “focus change requests” (*e.g.,* “Let’s make a car reservation” while in the middle of making a flight reservation). Error corrections can be made via “change requests” or “clear requests”, which either modify or eliminate data in the tree, respectively. The pragmatic module handles this domain act by effecting the appropriate change in the semantic system.

The pragmatic analysis module examines the user input and determines if a user explicitly answered a yes/no question, or provided expected, unexpected, or unrelated AV pairs in response to the system prompt. The **pragmatic scoring** module uses the results from this user utterance classification and the confidence associated with raw attribute-value pairs to update the pragmatic score of each of the corresponding candidate values. For example, a “neither” response to an explicit value ambiguity disambiguation question, (e.g., “Are you leaving from Atlanta or from New York?”) is taken as strong evidence against both values. This is critical for handling ambiguity in the user input. Given multiple candidates for a given attribute, we update the pragmatic confidence score for each candidate using MYCIN style formulas [8, 1]. The MYCIN algorithms for combining evidence for or against a candidate result in confidence factors s that range [-1, 1] for every candidate and are tunable to reflect how dramatic the change of confidence will be in the presence of new evidence[1].

[1]The MYCIN algorithm for combining evidence was preferred to more modern statistical methods (e.g., graphical models) due to its simplicity. A full-blown Bayesian approach would have required estimates of prior distributions over all of the attributes. Furthermore, the non-independence of attributes in any given system would have required many joint (or conditional) probability distributions. Since the training data of system interactions are very sparse, it is infeasible to calculate these probability distri-

2.3. Dialogue Manager

The main functionality of the dialogue manager is to elicit attribute-value pairs from the user, to resolve value and position ambiguity, to inform the user about updates in the application tree, to perform database queries and communicate results to the user, and to help the user navigate the database results. The main components of the dialogue manager are the electronic form, the agenda and the adaptive initiative modules[2].

The electronic form (e-form) consists of a set of attributes that are needed to form a database query (similar to a traditional e-form), and scores associated with those attributes, e.g., a flight leg is an e-form consisting of departure/arrival city, time, and date attributes. A score in the range [0,1], indicating the importance of filling a particular attribute, is attached to each attribute in the application tree and is dynamically updated at each dialogue turn. For example, for a flight query, cities and dates are more critical than time or airline carrier information. Conditional importance of attributes is also captured in the e-form, e.g., when the arrival date is specified, the importance of specifying a departure date is reduced.

The agenda consists of a sequence of e-forms and domain acts. Examples of e-forms are flight legs, hotel reservations, and car rentals. Typical application actions include summarization of flight information, confirmation of e-form values, and database query. A focus or context is associated with each entry in the agenda, e.g., 'first flight leg' can be the focus of an e-form agenda entry. There is also machinery to add/delete parts of the agenda as per user requests.

2.4. Natural Language Generation

Spoken responses to the user are handled by the natural language generation module. Typically, the system decides on a number of dialogue actions that need to be communicated (e.g., implicit value confirmation, explicit questions, or disambiguation dialogues), which cause the selection of one or more templates that are filled with values from the application tree. For example, "confirm city" and "request time" might select the template "What time do you want to leave *city*?" The actual forms of the templates, as well as the requested actions are determined by the adaptive initiative module.

butions directly from data. The proposed algorithm has the advantage of being able to integrate any kind of evidence (for or against). In addition, there are few parameters to train (and thus few training data are required) and the resulting confidence scores do not depend on the order in which evidence is considered.

[2]The adaptive initiative tracking algorithm used in the Communicator is described in [4].

3. MULTIMODAL COMMUNICATOR

The Communicator spoken dialogue system has been designed to be domain and modality-independent: adding the visual modality thus required only a few enhancements that proved easy to design and implement. We describe next how the semantic and pragmatic modules where updated and how the graphical user interface (GUI) was built. The visual modality consists of pen and graffiti input (or keyboard and mouse input in a desktop environment), and text and graphics as output.

3.1. Semantic Module

The semantic representation used for both the visual and speech interfaces is the same. The semantics of the application, the prototype tree, are unchanged since the visual interface has the same functionality as the voice interface. The notepad tree and application tree now encode the instantiated *joined* semantics of the visual and voice modalities.

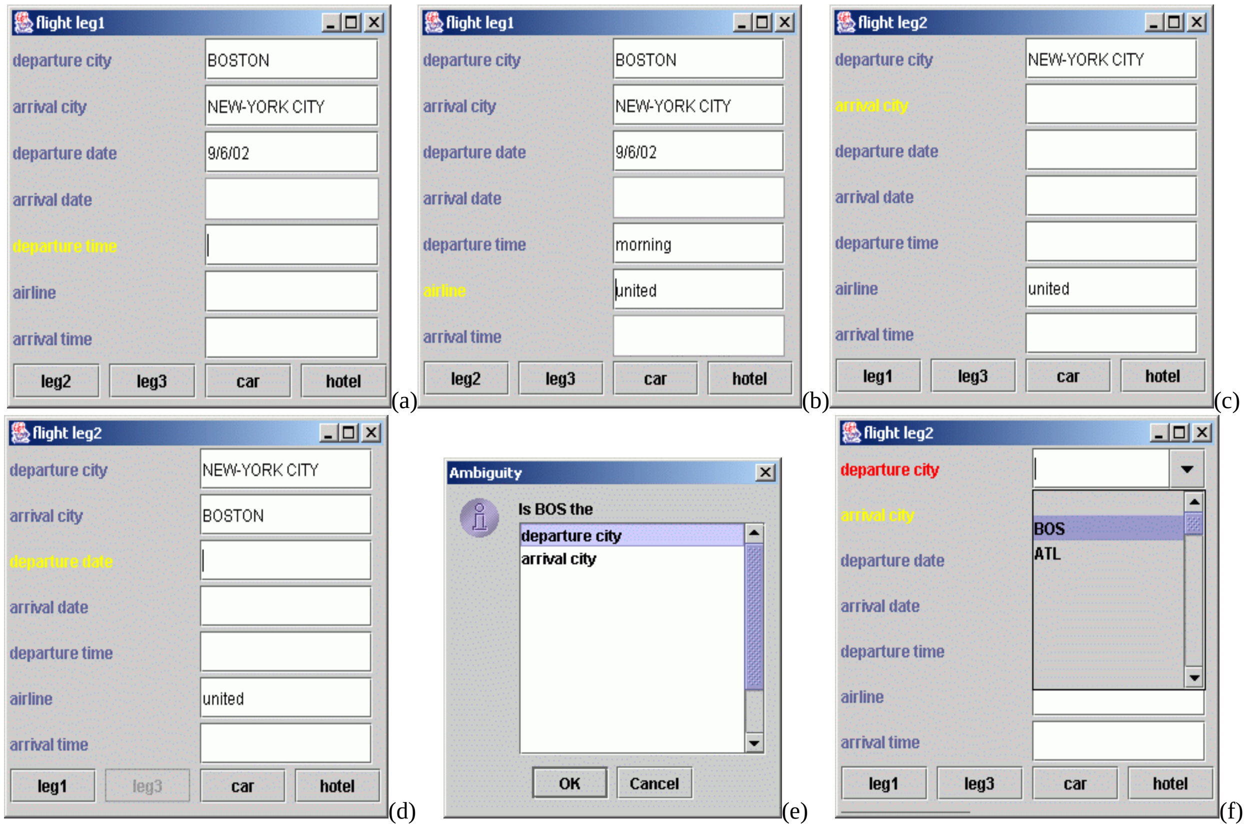
The GUI parser is the recursive finite-state parser used in the unimodal Communicator with an augmented grammar. In addition, to spoken forms the GUI parser understands abbreviations such as "10/3/02" for date or "15:00" for time.

3.2. Pragmatic Module

The context tracking, domain act classification and pragmatic analysis algorithms developed for the speech modality are also used for GUI input, although, in practice, only a small subset of their functionality is exercised. The pragmatic scoring algorithm is also unchanged. It has been designed to allow integration of evidence for or against candidate values and thus provides a very useful framework for merging often conflicting information collected from the two input modalities: given multiple candidates for a given attribute, we update the pragmatic confidence score for each candidate using MYCIN style formulae as before. The only difference is that now the initial confidence scores for values entered via the GUI are much higher than corresponding values provided from the recognizer. Similarly, the ambiguity detection, representation and resolution algorithms are the same for the both the unimodal and the multimodal Communicator. As originally intended, the Communicator pragmatic module design proved to be a natural and efficient way of combining information from different input modalities.

3.3. Graphical User Interface

The GUI forms are generated automatically from the prototype tree, the e-form definition and the agenda in the Com-


(a) flight leg1

departure city	BOSTON
arrival city	NEW-YORK CITY
departure date	9/6/02
arrival date	
departure time	
airline	
arrival time	

leg2 leg3 car hotel

(b) flight leg1

departure city	BOSTON
arrival city	NEW-YORK CITY
departure date	9/6/02
arrival date	
departure time	morning
airline	united
arrival time	

leg2 leg3 car hotel

(c) flight leg2

departure city	NEW-YORK CITY
arrival city	
departure date	
arrival date	
departure time	
airline	united
arrival time	

leg1 leg3 car hotel

(d) flight leg2

departure city	NEW-YORK CITY
arrival city	BOSTON
departure date	
arrival date	
departure time	
airline	united
arrival time	

leg1 leg3 car hotel

(e) Ambiguity

Is BOS the
departure city
arrival city

OK Cancel

(f) flight leg2

departure city
arrival city
departure date
arrival date
departure time
airline
arrival time

BOS
ATL

leg1 leg3 car hotel

Fig. 2. Graphical User Interface after the following input sequence (a) Speech input: "I want to fly from Boston to New York city on September sixth", (b) GUI input: "morning", "united" in the corresponding fields, (c) GUI input: 'leg2' button pressed, (d) Speech input: "round-trip" (e) Example of graphical representation of position ambiguity that has arisen from the speech interface; it is unclear if Boston is the arrival or departure city for this leg. (f) Example of graphical representation of value ambiguity; it is unclear if Boston or Atlanta is the departure city for the second leg.

municator dialogue manager. The e-form that generates the GUI form shown in Fig. 2 is:

```
travel.flight.leg1.departure.city 1.0
travel.flight.leg1.departure.date 0.9
travel.flight.leg1.departure.time 0.7
travel.flight.leg1.arrival.city 1.0
travel.flight.leg1.arrival.date 0.8
travel.flight.leg1.arrival.time 0.5
travel.flight.leg1.airline 0.6
```

Note that the fields in the GUI are ordered according to the e-form score that encodes the necessity of an attribute for successfully completing the query[3]. The buttons at the bottom of the GUI represent all the possible e-forms that are accessible to the user. This information is extracted and dynamically updated from the agenda. Value ambiguity is shown as a pull-down box with a list of choices and highlighted in red; position ambiguity and error messages as represented in the GUI as pop-up windows. Fields and buttons that become inaccessible in the course of the interaction are "grayed out". Finally, the context (or focus) of the interaction is highlighted in yellow. More information about the design of the multimodal user interface can be found in the next section.

[3]It is arguable if the grouping should be done by e-form score or by semantic coherence, e.g., all 'departure' fields grouped together followed by all 'arrival' fields. Clearly inserting the 'airline' field between 'departure time' and 'arrival time' fields seems awkward.

4. MULTIMODAL USER INTERFACE

A fundamental issue when designing multimodal systems is the choice, integration and appropriate mix of input and output modes in the user interface [9, 11]. Few guidelines exist for selecting the appropriate mix of modalities [2, 3], however, it is clear that a spoken language interface is not always the best choice. This is especially true for system output, where speech plays a secondary role to visual input (with the exception of hands-free applications). It is often the case when designing multimodal user interfaces, that the developer is biased either towards the voice or the visual modalities. Our goal is to follow an approach that respects both modalities, creating an interface that is both *natural* and *efficient*. The first step towards the seamless integration of the input and output modalities is a *consistent* user interface: the GUI represents system state and possible actions (including state and actions that are specific to the voice-modality). A second important step towards a truly multimodal experience is creating a user interface that utilizes the *synergies* between the input and output modalities, e.g., using visual feedback from the recognizer (including possible ambiguities).

Consistency is maintained via the common semantic representation, the use of e-forms as the building block for the application manager for both modalities, and the multimodal pragmatic scoring algorithm as described in the previous section. In addition, the same system functionality can be accessed by the user either via the speech or the visual modalities. More importantly, the modalities have been integrated in a balanced and synergistic way that allows the user the freedom to pick and choose the modality of preference. A short list of synergies between the speech and visual modalities as manifested in the multimodal Communicator follows: a) the system semantic state is represented visually, b) speech prompts are thus significantly shorter; mostly used to emphasize visual information, c) speech recognition errors are easily corrected via the GUI, d) position and value ambiguities are displayed visually and are easily resolved via the GUI (see Fig. 2(e),(f)), e) the focus (or context) of the dialogue is high-lighted visually and can easily be changed via the GUI, f) GUI takes full advantage of speech-interface "intelligence", g) conflicting GUI and speech input are seamlessly integrated via the pragmatic scoring algorithm. Examples of the seamless integration between the visual and speech modalities are shown in Fig. 2. In Fig. 2(a), the semantic state is displayed visually; attribute-value pairs ("departure city", "Boston"), ("arrival city", "New York City") and ("departure date", "9/6/02") are derived from the user input. In addition, the "arrival date" field is disabled since the "departure date" is specified by the user (using the intelligence of the speech interface), and the focus of the next dialogue turn, i.e., "departure time", is highlighted. In Fig. 2(b), similar behavior can be seen but this time the GUI is used for input. In Fig. 2(c), the values for the departure city for the second leg of the trip and the preferred airline are deduced based on the first leg information. In Fig. 2(d), the user informs the system that this is a round-trip: the "arrival city" is deduced and the "leg3" button is disabled. In Fig. 2(e) and (f), the user is prompted to resolve position and value ambiguity respectively. The ambiguity has appeared in the speech input, however, the system represents ambiguity graphically and suggests pen input to the user as the more efficient and natural modality.

4.1. Dominant Modality and Modality Tracking

Turn-taking in a multimodal user interface can easily become a headache. The dynamics of turn-taking and the latency of the interfaces are very different for the speech and visual modalities. These problems are commonly tackled either by adopting a "click-to-talk" approach, where the user has to press a "speech-activation" button (often tied to specific application semantics) to activate the speech recognizer [7], or by an "open mike" approach where the speech modality is always available [9]. In a "click-to-talk" scenario, the visual modality is often dominant, while in an "open-mike" scenario, the speech modality dominates the interaction. In practice, "click-to-talk" is more commonly used both because it is easier to implement[4] and because the visual interface is more efficient (especially in desktop applications).

In the multimodal Communicator, we introduce an interface design that is a mixture between the "click-to-talk" and "open-mike" approaches. At each point in the interaction the system determines the **dominant modality** be it speech or visual. The dominant modality is computed from two sources of information: the status of the interaction (dialogue act and expected user input) and past user behavior. For example, when the system expects the user to select information from a pull-down menu the visual modality is more prominent (and thus the user has to "click-to-talk"), while for entering free text the speech modality is more probable to be used ("open-mike"). Past user behavior also conditions the modality tracking algorithm; for users that have shown preference towards GUI input (and thus have overridden the selection of the modality tracking algorithm) the visual modality will be chosen more often as the dominant one. Suppose that the system expects input (type) i; the modality of choice $\hat{m}$ for user u is given by the Bayesian formulation [5]

$$\hat{m} = \arg\max_m P(m|i,u) = \arg\max_m P(m|i)\frac{P(m|u)}{P(m)} \quad (1)$$

where $P(m|i)$ is the probability of a stereotypical user selecting modality m for input type i, $P(m|u)$ is the probability that user u will select modality m, and $P(m)$ is the a priori probability that a stereotypical user will select modality m. In the above formula, $\frac{P(m|u)}{P(m)}$ is a dialogue-independent term and expresses the bias that user u has towards modality m compared to the average user, while $P(m|i)$ is a dialogue-dependent term that computes the expected modality based on the type of input the system is

[4] In an "open-mike" scenario the latency of the speech interface may cause the system to fall out-of-turn with the user. To resolve this problem "blocking" is implemented: the visual interface is inactive while the speech input is being processed by the system and vice versa.

[5] To arrive to Eq. (1) we assume that u and i are independent variables.

expecting. In our system, i encodes the linguistic complexity (aka perplexity) of the expected input, e.g., if the user is expected to specify the departure city the perplexity is (approximately) equal to the number of airports in the database. The higher the perplexity of the expected input i the higher $P(m = \text{speech}|i)$ is. Maximum likelihood estimates of the distributions $P(m)$, $P(m|u)$, $P(m|i)$ can be easily computed from (automatically annotated) multimodal dialogue data.

Based on the adaptive modality tracking algorithm the multimodal spoken dialogue interface displays the following behavior: when the speech modality is dominant the system will play a speech prompt and wait for input; the user can override the modality selected by the system by using the GUI (in which case prompt playback stops; GUI "barge-in"). When the visual modality is dominant the system does not play a speech prompt and waits for GUI input; the user may again override the system choice by pressing on the speech input activation button. The adaptive modality tracking algorithm tries to predict the user's preferred modality at each point in the interaction based on interface efficiency considerations and user preferences. We believe that this novel interface design based on adaptive tracking of the dominant modality significantly improves the quality of the interface by increasing the naturalness and flexibility of the interaction. More research and formal evaluation is needed to verify these claims.

5. DISCUSSION

Informal evaluation of a multimodal prototype travel reservation application provided positive feedback for the interface design and system functionality. Users appreciated the naturalness and flexibility of the interface and were able to complete the interaction more efficiently than with the unimodal system. Formal evaluation of the prototype remains to be done. Further research is needed to tune the adaptive modality selection algorithm to maximize user satisfaction.

Acknowledgments: The authors would like to express their sincere appreciation to Manolis Tsangaris, Shri Narayanan for many helpful discussions and to Jeff Kuo, Andy Pargellis, Joe Olive, Chin-Hui Lee for their many contributions to the Bell Labs DARPA Communicator program.

6. REFERENCES

[1] E. Ammicht, A. Potamianos, E. Fosler-Lussier, "Ambiguity Representation and Resolution in Spoken Dialogue Systems," in *Proc. EUROSPEECH*, (Aalborg, Denmark), Sep. 2001.

[2] N.O. Bernsen and L. Dybkjaer, "Is speech the right thing for your application?," in *Proc. ICSLP*, (Australia), Oct. 1998.

[3] V. Bilici, E. Krahmer, S. te Riele, R. Veldhuis, "Preferred Modalities in Dialogue Systems," in *Proc. ICSLP'2000*, (Beijing, China), Oct. 2000.

[4] J. Chu-Carroll, "Formbased reasoning for mixed-initiative dialogue management in information-query systems," in *Proc. EUROSPEECH*, (Budapest, Hungary), Sept. 1999.

[5] P. Cohen, M. Johnston, D. M. and S. Oviatt, J. Clow, and J. Smith, "The efficiency of multimodal interaction: A case study," in *Proc. ICSLP*, (Sydney, Australia), 1998.

[6] M. Galley, E. Fosler-Lussier, and A. Potamianos, "Hybrid natural language generation for spoken dialogue systems," in *Proc. EUROSPEECH*, (Aalborg, Denmark), Oct. 2001.

[7] X. Huang et al,"MIPAD: a next generation PDA prototype," in *Proc. ICSLP*, (Beijing, China), Oct. 2000.

[8] D. Heckerman, "Probabilistic interpretations for MYCIN's certainty factors," in *Uncertainty in artificial intelligence* (L. Kanal and J. Lemmer, eds.), North Holland, pp. 11–22, 1986.

[9] S. Narayanan and A. Potamianos, "Creating conversational interfaces for children," in *Trans. on Speech and Audio Proc.*, vol. 10, no. 2, pp. 65-78, Feb. 2002.

[10] S.L. Oviatt et al, "Designing the user interface for multimodal speech and gesture applications: State-of-the-art systems and research directions," Human Computer Interaction, vol. 15, no. 4, pp. 263-322, 2000.

[11] A. Potamianos et al, "Design principles and tools for multimodal dialog systems," in *Proc. ESCA Workshop Interact. Dialog. Multi-Modal Syst.*, (Kloster Irsee, Germany), June 1999.

[12] A. Potamianos, E. Ammicht, and H.-K. Kuo, "Dialogue management in the Bell Labs communicator system," in *Proc. ICSLP*, (Beijing, China), Oct. 2000.

[13] A. Potamianos and H.-K. Kuo, "Speech understanding using finite state transducers," in *Proc. ICSLP*, (Beijing, China), Oct. 2000.

[14] M. Tsangaris and A .Potamianos, "AGORA: A GUI approach to multimodal user interfaces," in *Proc. HLT*, (San Diego, California), 2002.

[15] Q. Zhou, A. Saad, and S. Abdou, "An enhanced BLSTIP dialogue research platform," in *Proc. ICSLP*, (Beijing, China), Oct. 2000.