3

Multimodal Integration in Embodied Conversational Kiosk with Augmented Reality

by

Jane H. Huang

Submitted to the Department of Electrical Engineering and Computer Science
In Partial Fulfillment of the Requirements for the Degrees of

Bachelor of Science in Computer Science and Engineering

and

Master of Engineering in Electrical Engineering and Computer Science

at the
Massachusetts Institute of Technology
May 24, 2001

Signature of Author [illegible]
Department of Electrical Engineering and Computer Science
May 10, 2001

Certified by [illegible]
Professor Justine Cassell
Professor of Media Arts & Sciences
upervisor

Accepted by [illegible]
Arthur C. Smith
Chairman, Department Committee on Graduate Thesis

Multimodal Integration in Embodied Conversational Kiosk with Augmented Reality

by

Jane H. Huang

Submitted on May 10, 2001 to the Department of Electrical Engineering and Computer Science In Partial Fulfillment of the Requirements for the Degrees of Bachelor of Science in Computer Science and Engineering and Master of Engineering in Electrical Engineering and Computer Science

ABSTRACT

A multimodal integration module is designed and implemented for **M**edia Lab **A**utonomous **C**onversational **K**iosk (MACK). MACK is an embodied conversational kiosk that has an augmented reality paper map as part of its user interface. In this study, we examined several multimodal systems as well as important techniques in designing multimodal architectures. We also reviewed powerful multimodal integration frameworks such as Semantic Frame Merging, Unification, and Finite-State, and have designed MACK's integrator based on the technique that would maximize mutual disambiguation of speech and pen-input. The final integrator is built based on Johnston's finite-state transducer (FST) approach. Considering the context of MACK, we felt that the FST approach offered the most simple, cost-effective, and scaleable solution.

Thesis Supervisor: Justine Cassell

Title: Professor of Media Arts & Sciences

Acknowledgments

I thank Professor Justine Cassell for giving me the opportunity to work on this project and providing the much-needed guidance. She has taught me much about research, persistence, and power of determination. Her feedbacks, while harsh at times, have always been insightful.

I thank the members of GNL for making this year of research such a fun learning experience. In particular, I am grateful to Yukiko Nakano for her guidance and support, and to Linda Lin for her consistency and optimism.

I thank my all my dear friends and family for their love and support. Their belief in me has been the driving force through this last stretch. I owe my sanity to them.

To my parents for their love.

Chapter 1

Introduction

Since Bolt's demonstration of "Put-that-There," the area of multimodal systems has undergone rapid growth. Multimodal architecture has been incorporated in applications within various contexts, including Air Force Command and Control systems [14], personal appointment calendar [27], a military simulation system [9, 10, 12], and company directory access system (Multimodal Messaging System). Regardless of application context, a multimodal interface invariably facilitated more natural user interactions, while respective multimodal integration techniques have consistently shown improved task-performance. The collective success of these different multimodal systems provides the incentive for us to explore the potentials of multimodal integration in an entirely different domain. Specifically, we would like to implement an efficient multimodal integration framework for an embodied conversational kiosk with an augmented reality paper map; that is, the paper map extends the map from the virtual world into the tangible world [3].

An embodied conversational kiosk (ECK) is an interactive kiosk with a graphical representation. The particular system of interest is the **M**edia Lab **A**utonomous **C**onversational **K**iosk (MACK). MACK is an ECK that has an augmented reality paper map as part of its user interface. The spatial model represented in each page of the paper map is stored in MACK's knowledge base, such that the spatial representations of the building are consistent between the real and virtual world. The consistency, in turn, provides a seamless integration of the user's reality and MACK's reality as the user is now able to indicate to both himself/herself and the kiosk on the same map [3]. From this interface, MACK accepts spoken inputs and pen-inputs made on the map. Consequently, the multimodal integration for MACK involves integrating speech and pen inputs.

In Chapter 2, we study previous work related to the development of multimodal systems, from property analysis of multimodal architectures to empirical studies on multimodal interactions, and from past systems to prevalent integration theories. Chapter 3 analyzes the issues specific to the current system. We examine the properties that distinguish the kiosk (with shared reality) domain from other system domains and restate the specific objectives of this multimodal integration framework. In addition, we discuss the integration of a framework to our kiosk. Lastly, we will also present performance evaluation on the implementation. Following the discussion on current implementation, we propose potential future expansions of the system in Chapter 5.

keywords:

mutual disambiguation (MD), unification, typed feature structures, finite-state transducer (FST), typed categorization.

Chapter 2

Related Work

Before we can begin our implementation of a multimodal integration algorithm, we need to obtain a good knowledge about the basic properties of a multimodal architecture. In the next section, we examine empirical studies that have shown insights on some inherent strengths and weaknesses of each approach. We follow the empirical studies with brief discussions of previous systems and their relevance to our objective. Finally, we discuss three current multimodal integration frameworks.

2.1 Literature review

2.1.1 "Why multimodal?"

A gross simplification may render the answer, "Because it's cool, and we have the technology to support it." But does the pursuit of technological complexity and or the desire to create cool gizmos explain the emergence and growing popularity of multimodal systems?

The answer here is a sound "no." Multimodal systems are more than just a combination of different technologies. Instead, the integration of multiple modalities can have synergistic effects on the overall perceived information [17, 18]. Empirical studies have shown that the flexibility of input modalities permits the following:

1. the accessibility for diverse users and usage contexts;
2. increased stability and robustness as measured by performance; and
3. enhanced expressiveness and efficiency [19].

It is easy to see that the option of multiple input modalities can accommodate various users' strengths and preference, and thus can increase the "naturalness" and the comfort level during a user's interaction with the system. Meanwhile, it is also worthwhile to note the unique capacity of individual modalities. That is, different modalities can "differ in the type of information they transmit, their functionality during communication, the way they are integrated with other modes, and in their basic suitability to be incorporated into interface styles"[18]. For example, graphical inputs are better for tasks involving spatial orientation, while spoken inputs are likely more receptive to represent hierarchical information. Similarly, various input modalities may be more suitable for use in different environments [19]. It is important to note that multimodal systems are valuable indeed not only because it accepts inputs of multiple modalities but also because it relies on them. Many people mistakenly assume that systems that include speech as an input modality rely on speech to be the primary input, but empirical studies have shown that different input modalities are used for different purposes [18]. Conversely, information transmitted through different input modalities is not redundant [18]. Meanwhile, the different capacities together again echo the theme that complementary modalities could better span the communication spectrum of a natural interaction. *To the end of enabling more natural interactions, we see that multimodal systems are preferred.*

While the flexibility offered by multimodal systems may be self-evident, the correlation between multiplicity of input modalities and performance evaluation is subtle. Empirical studies have shown that multimodal systems provide an inherent advantage in error handling [17]. At the mechanical level, the possibility of input-mode alternation at least prevents the "overuse and physical damage to any individual modality during extended periods of use"[19]. Moreover, according to Oviatt, users do not always interact multimodally for all tasks [18], but instead would select the input mode that they deem to be less "error prone" in delivery [19]. For example, as users feel that speech is the most reliable way to deliver general action commands, they would naturally prefer interaction unimodally (through speech only) such action commands as "specify constraint," "overlays," "locate," "print,". Conversely, as users perceive that speech alone is not sufficient to deliver spatial information, they naturally lean towards multimodal interaction for tasks involving a spatial location description [16]. Such intuition in users facilitates error avoidance naturally. Furthermore, other studies have shown that multi-modal systems help induce spontaneous error-recovery. For instance, users are inclined to switch input modes after encountering system errors [19]. In other words, if the user feels that his/her speech command is not understood by a multimodal system, he/she would attempt to issue the command through a different method. By facilitating alternative input modes, multimodal systems allow the user the option to alternate input modalities, and thus facilitate error-recovery naturally. Last but not least, studies have also shown that users are subjectively less frustrated with errors when interacting multimodally, as opposed to unimodally, even though error-occurrence may be same for both types of interactions [19]. The reduction in frustration level may be directly attributed to the fact that multimodal systems, by allowing their users to interact differently, permit the users more room for error-recovery. *With the objective of reducing user frustration, we also conclude that multimodal systems are preferred.*

On the subject of error handling, multimodal systems have been shown to successfully support significant levels of mutual disambiguation [17]. Mutual disambiguation describes the recovery of unimodal recognition errors within multimodal architecture [17]; in other words, accepting the inputs from all the separate modalities (assume the performance of each input device to be independent) in an optimization algorithm to determine the most-probable interpretation of the user input. In an empirical study using Quickset, a multimodal system that adopted a unification-based integration algorithm, the speech recognition error rate is decreased by 41% when the raw signals are recognized and interpreted in parallel with one or more other modalities [17, 18, 19]! This data provides both irrefutable evidence on the synergistic powers inherent in and the strategic importance in the design of multimodal systems. At a time when no sensory input tools, such as speech-, gesture-, or pen- recognition devices are 100% error-proof, mutual disambiguation is a design-bonus too great to be ignored. *Relating this error-reduction tool to prior arguments, the potential of mutual disambiguation becomes yet another reason for multimodal systems over unimodal systems.*

Finally, the increased efficiency with multimodal systems is also remarkable. According to Oviatt [19] multimodal interaction using speech and pen in the Quickset system was shown to be nine times faster than using a graphical interface for the same task. Moreover, a different study also using voice/pen study has demonstrated a 10% faster task completion time [15].

2.1.2 "In particular, why is a second modality good for a conversational system?"

In addition to the increased performance of a generic multimodal architecture, it is worthwhile to discuss the particular advantages that a multimodal architecture would provide to a system that uses speech. A notable convenience would be the ability to resolve deictic terms in speech using inputs from other modalities [1, 14]. Deictic terms are demonstrative pronouns used to specify extra-linguistic referents, such as "this," "that," "here," and "there." As the referents of these deictics often rely "entirely on the circumstances, " [24] researchers and engineers have depended on the explicit pointing gestures that often accompany these terms to identify the referent. However, deictic resolution is neither the more frequent nor the most significant achievement to multimodal systems. Linguistic analysis on empirical data from a simulated dynamic map system shows that more than 59% of multimodal utterances did not contain a spoken deictic term [16]. The same study has also shown that multimodal pen/voice language less prone to disfluencies and contains less content errors [15]. That is, during a multimodal interaction that includes speech and another input modality, user-data have shown a significant decrease in the rate of speech disfluencies, phenomena such as content self-corrections, false starts, verbatim repetitions, filled pauses, and self-corrected spellings and abbreviations in the users' utterances [15]. In theory, disfluency is a sensitive measure of cognitive load during human-computer interfaces. Therefore, the reduction of disfluency through the design of a multimodal architecture presents exciting potentials of multimodal systems to improve the human-computer interface. On the other hand, the linguistic simplification facilitated by multimodal systems would also support faster task-completion times as well as less computational processing of the language itself [15].

2.1.3 Observations in Multimodal Systems

In addition to improved performance, empirical studies have unraveled insights important to the design and implementation of a multimodal kiosk. One important observation is the synchronization of input modes. Contrary to what intuition may suggest, not all input signals during a multimodal interaction are simultaneous [16, 18]. An empirical research of pen/voice human-computer interaction has confirmed that temporal precedence of written input is a consistent theme [16]. Specifically, the study has shown that the onset of a pen input preceded deictic terms 100% of the time when the signals are sequential, and 60% when simultaneous; a sequential integration pattern is one characterized by having a distinct time lag between the inputs of different modes, and a simultaneous pattern is where the inputs overlap temporally [16]. Moreover, the linguistic analysis has shown that spoken deictic terms lag written inputs by 1.4 seconds on average [16].

The thematic precedence of written inputs is significant to the design of an integration algorithm because it is intimately related to the temporal linearity of input modes. We will return to this point later in the design discussion.

2.2 "How have they done it in the past?"

Although speech and pointing are not necessarily the dominant pattern of multimodal integration [18], speech-and-pointing systems have marked important milestones in the development of multimodal architectures. We present a few systems below to demonstrate the different objectives and domains in which multimodal systems have been implemented.

Put-That-There [1]

Bolt's "Put-That-There" was the first to combine voice and pen for the purpose of providing a "concerted, natural user modality" (p. 262). Put-That-There used one of the first "connected-speech" speech recognition engines[1], the DP-100 Connected Speech Recognition System, for its speech input, and a spatial orientation- and position-sensing technology, the Remote Object Position Attitude Measurement System, for gesture input observation. Together, the system allowed users to make commands about simple geometric shapes on a large display surface using both voice and corresponding pointing gestures. Some suitable commands that combine voice and pointing include

- "*create* {name of a basic shape} there"

[1] Earlier speech recognition tools used "discrete utterance" speech recogntion required the speak to the system in a "clipped" or "word-by-word" style (p. 264). The use of a connected-speech recognition engine allowed further facilitation of natural environment for the user.

- "*move* {name of a basic shape, or "that" in conjunction with pointing}"
- "*copy* {name of a basic shape, or "that" in conjunction with pointing}"
- "*make* {name of a basic shape or "that" in conjunction with pointing }

  {transformation, either using specific modifiers, such as "smaller," or by referencing another item, using "like that," while indicating some other item through pointing}"
- "*delete* {name of a basic shape, or "that" in conjunction with pointing}"

However, the semantic processing would be based on the spoken command, and pointing gestures would be incorporated only to resolve deictic terms. In other words, only after a verbal command has been accepted and parsed, is a gesture input then considered for reference.

While the system presented a successful case of deictic referencing, its complete dependence on speech-input has severe consequences. For one, this design would not benefit from any mutual disambiguation from the two modalities. In fact, the "Put-That-There" architecture permitted little of the previously mentioned flexibility that may improve the system robustness and performance. Moreover, the constraint on using the non-speech input mode only in the context of the speech-input also does not fulfill the contextual coverage that these two complementary modalities would potentially be able to.

CUBRICON (CUBRC Intelligent CONversationalist) [14]

CUBRICON is an intelligent multi-media interface system that was developed as part of the Intelligent Multi-Media Interfaces (IMMI) project, where the focus was to "simplify operator communication with sophisticated computer systems" (p13). The domain here is the Air Force Command and Control (C2) system. The goal is to model and facilitate dialogues that are natural in human-to-human interactions through the integration of speech and pointing gesture inputs (through a mouse on a graphical display).

Like "Put-That-There," CUBRICON is a task-based system where the user is allowed to command over entities either by referring to their proper names or by pointing. Some acceptable multimodal commands include:

- "*what is* {proper name of an object, or "this" in conjunction with pointing}"
- "*display* {proper name of an object, or "this" in conjunction with pointing}"
- "*where is* {proper name of an object}"

The multimodal architecture in CUBRICON also used non-speech input modes to "accommodate" the speech input as necessary. The inclusion of pointing gestures was only for referent identification, while semantic interpretation remained contained within speech-input. Unlike Bolt's system, however, CUBRICON maintained particular interest supporting the naturalness of use-interaction. The CUBRICON system included discourse models as well as user models such that dialogue continuity and relevance is preserved. For example, CUBRICON maintains an "entity rating system" that keeps track of the relevance of the entity to that user. That is, a numerical rating of an entity is increased when the user mentions it; once the rating exceeds a critical threshold, CUBRICON would automatically label the entity as being critical to the user and/or the task. In turn, this information would allow CUBRICON to maintain a current-state information about the user.

Gandalf [20, 21]

Gandalf is a multimodal system created to "carry out embodied, topic-oriented dialogue" (p. 18) system through recognizing user's speech, gaze, and gestures. The system is noted for its ability to perceive and generate in response, multimodal behaviors such as deictic gestures (such as pointing or iconic gestures), attentional and deictic functions (through tracing gaze movements), prosody (through speech recognition and generation of back channel feedback or meaningful utterances), and turn-taking signals (through a combination of gaze and intonation). The integration of multiple input modalities, in keeping with the objective to better facilitate real-time decision making ability of a system, observed these following issues in design:

1. Multi-layered Input Analysis: since actions in different modes may overlap or have different temporal constraints, such reactive and reflective[2] responses must both be observed.
2. Temporal Constraints between behaviors. The structure of dialogue implies that certain behaviors are "expected to happen within a given time span," and that the violation of such timing constraint would alter the meaning of the action (p16).

[2] Reactive responses refer to behaviors that require recognize-act cycles shorter than 1 second, where reflective responses are those that require cycles longer than 1second (27).

3. Interpretation of inputs and dialogue state is fallible.
4. redundancies and deficiencies are both plausible in the sensory data.

Even though the use of pointing for deictic resolution is still incorporated in Gandalf, multimodal integration here is used to serve a different goal. Where Put-That-There and CUBRICON concentrated mainly on improving resolution of dialogue *content* by integration multiple communications channels, Gandalf is focused on enhancing overall *dialogue management*. Gandalf uses the combination of speech recognition, speech-intonation analysis, and gaze input to manage turn-taking in the dialogue.

2.3 Multimodal Integration Models

As empirical studies suggest that speech-input is often not the primary or exclusive carrier of content [18], it is evident that deictic pronoun referencing is no longer a sufficient multimodal integration framework. Since the completion of "Put-that-There" and "CUBRICON," more sophisticated multimodal integration frameworks have been developed which are less speech-oriented. Where "Put-that-There" and "CUBRICON" used non-speech inputs only for the purpose of enhancing or completing a speech-input interpretation, later frameworks strived to achieve balance between and mutual disambiguation of various input modalities. Some of the most effective techniques are semantic frame merging, unification-based, and finite-state transducer-based. Each framework is presented below, followed by the description of a prototype system which embodied the integration framework.

2.3.1 Semantic Frame Merging [27]

Multimodal integration through semantic frame merging was the first framework to incorporate a "uniform handling of high-level information" from various input modes (p3546). In this approach, the semantic interpretation of each input signal is represented as "a frame consisting of slots," where the slots encapsulate specific entity-types that could be indicated by more than one input mode. Combination of these partially filled frames, called "merging," then comprises the integration across the modes. Where merging partially filled frames during each input event would render a multimodal interpretation, merging with previous interpretation-frames would help retain contextual information about the interaction in its entirety. We present the specific mechanisms of the framework in the discussion of Jeanie below.

This methodology was a break-through in multimodal integration. Being the pioneer at uniform representation of semantic information, that is, generating the same high-level semantic representation of an input regardless of the input modality, it was the first framework to embrace modularity and scalability in a multimodal architecture. The capacity to "translate" various input signals to a common interpretation implies an ease of replacement as well as an ease in increasing the number of input modes. As we shall see in the following section, both the concept of common semantic representation and that of preserving a context history would become essential themes of multimodal integration.

System: Jeanie: A Multimodal Calendar [27]

Jeanie is a multimodal interface that accepted speech-, pen-based gesture-, and handwriting-inputs. Jeanie allowed users to perform appointment-scheduling functions through a mixture of its input modes. Jeanie uses the Phoenix semantic parser to parse the natural language inputs into fragments. By matching speech input to a previously determine set of grammar and vocabulary (based on a set of 128 utterances collected from user studies), skipping over unknown words and unmatched fragments, the parser outputs concepts for the speech frame and identifies the slot that need to be filled. Consider the scenario where the user says "I am not meeting with Sam" in conjunction with the act of crossing out "Sam" on the graphical display. The parser would identify a speech frame "delete meeting," a slot for a person_name from speech input, and a slot for a person_coordinate from the pen input (the items on display are recorded by the spatial location). Merging the frame and the slots, consequently, would generate the command, "delete meeting [with] Sam."

The preliminary performance evaluation of Jeanie showed the multimodal interpreter resolves input correctly up to 80% of the time. However, the data showed an 18% decrease in resolution accuracy when recognition errors occur either in speech or pen-input recognition. The result indicated that recognition errors from speech and pen-input had similar effects on the interpreter's resolution accuracy, and thereby suggested presence of cross-modal redundancy in the system. Whether or not such redundancy reflects Jeanie's system architecture or the effectiveness of the frame merging technique remains unclear.

2.2.2 Unification-based Multimodal Integration [9, 10, 12]

Unification is probably the most thorough approach taken towards integration of multiple modalities. The approach fully exploits the mutual disambiguation potential in a multimodal architecture in its two-step process: semantic pre-processing of individual input modes, followed by the unification of typed feature structures across the modes. A typed feature structure is similar to that of a "slot" in semantic frame merging ideology. It is the common representation of the "semantic contribution" of an input [p625, 5]. For example, the gesture, point at location A, has the same typed feature structure as the utterance, "location A." Similar to the effect of "slots" in semantic frame merging, the use of typed feature structure promoted better multimodal interpretation as it allowed a full command to be in either mode. However, typed feature structure is a more generic and inclusive superset of "slots." Where "slots" are limited to particular type entities that can be indicated by *all* input modes present in the system, typed feature structures are simply any entities indicated in *any* input mode. The input signals from each mode would be processed for semantic interpretation, and a list of n-best guesses would be generated about that input signal based on pre-existing statistical model of the input device. That is, each semantic interpretation of the input is assigned a correctness probability estimate.

During a unification operation, typed feature structures with the highest correctness estimates are checked for consistency; compatible typed feature structures are combined while incompatible ones are filtered out. The set of semantic interpretations, as a result of unification, then, would each bear a correctness estimate that is the product of correctness probabilities assigned to the fused typed feature structures [9]. Unification is processed using a multidimensional chart parser [10] and can occur at two levels: within each input mode and across all input modes. For example, after a natural language parser has parsed a particular speech stream and labeled various components of the speech stream as different typed feature structures, unification may take place for speech-only input. If the typed feature structures identified within the speech input were consistent, and collectively comprise an acceptable command, then the unification operation within the speech mode would succeed, and a complete semantic interpretation of the speech input would be generated. Conversely, if the unimodal input lacked certain typed features, then unification within that input mode would result in a partial semantic interpretation. In this case, the multimodal unification would allow a new interpretation to be concluded from comparing typed feature structures from multiple modes. On the other hand, if a complete interpretation can be generated from a unimodal unification, a sequential multimodal unification operation is still available to confirm (or potentially increase) the correctness of interpretation. The final semantic interpretation in this approach, is the one with highest correctness estimate after unification.

In theory, unification approach is the most modular and flexible approach in multimodal integration. As it provides independent interpretation for each input mode these initial unimodal interpretations permit flexibility in generating a final interpretation. While unification recursively searches for an optimized interpretation, the n-best list is always preserved for further iterations of the unification operation. In other words, if structure A were unified with B during an iteration, the structure A_B would be assigned the joint probability of A and B, but structures A and B and their respective probability estimates remain intact for future unification with some unused structure C. Even if A and B cannot be unified, these structures remain uncorrupted and eligible for a next unification attempt. Therefore, no feasible interpretation would ever be over looked. Moreover, the preservation of unimodal interpretations would permit a "plug-and-play" [p. 363, 12] trend in replacing or adding input modalities into the architecture.

The flip side of the expressiveness, however, is the high cost of maintenance. In practice, the unification approach can harbor dangerous exponential growth amidst the feasible possibilities within each mode and among combinations of all modes. Furthermore, in order to assign proper probabilities, and to verify status of interdependence across the modalities, a mammoth amount of empirical research with real users must be conducted for statistical analysis. In particular, meticulous attention must be paid the interdependence of the input modes. Speech and gesture, for example, are highly correlated. Hence, the probability estimate of a multimodal interpretation is not always fairly represented by a joint probability; instead, a careful normalization process may be needed to adjust for the interdependence of input modes. We will revisit this point in greater details in the analysis of QuickSet.

Moreover, since the critical value in this methodology is the combined correctness probability estimate, the interpretation with maximum correctness estimate is not necessarily be one that contains all the maximally correct typed feature structures. In fact, it may not even contain *any* of the maximally correct typed feature structures. The statistical basis for correctness here only suggests that the final accepted interpretation is the most-likely interpretation, *on average,* and based on *normative* distribution. More importantly, the deduction used in finding a best solution is heavily reliant on the accuracy of the statistical model used.

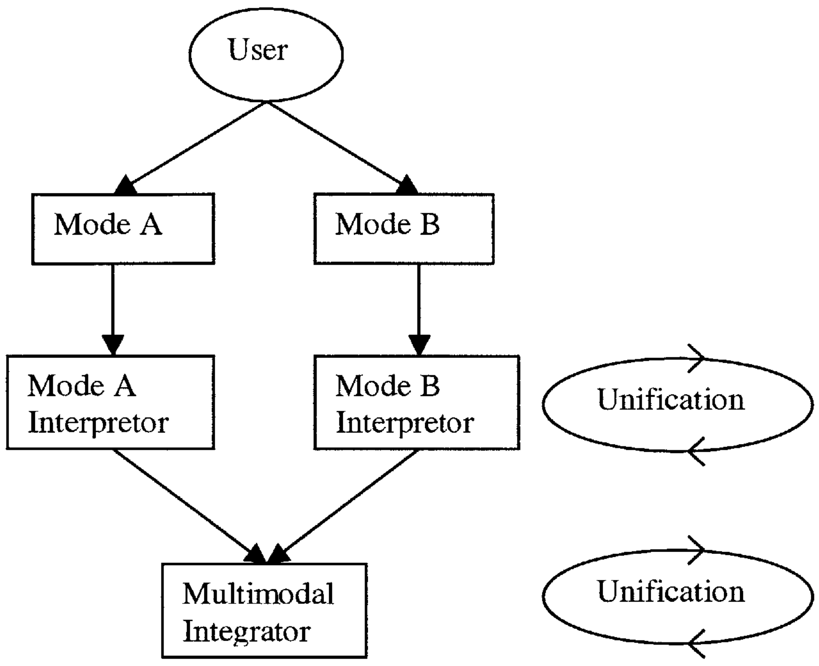

Figure 1. Unification-based Multimodal Integration Architecture

System: QuickSet: Multimodal Interaction for Simulation Set-up and Control [9, 10, 12]

Unification-based multimodal integration was first implemented in QuickSet, a pen/voice system developed for the training of US Marine Corps platoon leaders. The user interface to QuickSet includes a microphone (which can operate on a click-to-speak or open mode) and a map display of the terrain where the military simulation is to be run. During an interaction, the user can issue spoken commands while gesturing and drawing on the map simultaneously.

Using the unification-based approach, inputs from each mode are tagged with time stamps and are first unified unimodally. A list of n-best possible results is generated from the operation. If the list consisted of all executable commands, then the input is marked complete and does not need to be integrated multimodally. Otherwise, the input is tagged as partial and their time stamps are examined. The integrator would unify speech and gesture inputs if the respective active time windows overlap; the active time window rule observes that pen input typically precedes speech by 3 to 4 seconds [16]. In other words, if speech- and gesture-inputs are temporally compatible, then multimodal unification operation would proceed.

The statistical models used to compute the correctness probability value for multimodal unification results have changed since the original implementation of QuickSet. The original system relied on a mode-independence assumption [9]. Since speech and pen-gesture have been found to be highly correlated, QuickSet has experimented with various statistical models to approximate the relationship between speech and gesture. While no empirical results are available to evaluate the original QuickSet (with the mode-independence assumption), empirical results using different approximations have shown up to 95.26% correct using the Members-Teams-Committee[3] (MTC) approximation model [19, 23].

In the most general terms, QuickSet has delivered the flexibility and expressiveness that the unification approach had promised on paper. However, it has also demonstrated the inevitable complexity with respect to combinatorial growth and temporal restriction. Moreover, an empirical study experimenting different statistical

[3] MTC is a new recognition technique used by complex pattern recognition systems. The details of the MTC architecture is unnecessary for the current discussion, but may be reviewed in "Multimodal Integration ---A Statistical View," (28).

models with QuickSet has revealed a significant correlation between unification accuracy and the underlying statistical model [28]. Statistical models are derived from analysis of assumptions and data mining of empirical usage-data. This dependent relationship between accuracy and statistical model, presents yet another potential source of complexity.

On a completely different topic, QuickSet is a task-based system [9], meaning that it assumes the user is sufficiently knowledgeable of the information maintained by the system, and that the system performs particular tasks at the user's request. The central objective of a task-based system, then, is to increase resolution accuracy from the inputs such that a command could be carried out, QuickSet has little incentive about conflict detection. In other words, QuickSet assumes that the user is informed and thus always tries to "make sense" of the user's input instead of detecting conflicts/errors with the user input. Conflict detection in user inputs is a concept that we shall revisit shortly.

2.2.3 Finite-state Multimodal Parsing and Understanding [11]

The computational complexities of the unification-based approach precipitated the development of a "simpler" framework. This alternative approach attempts to use a weighted finite-state machine to parse, understand, and integrate multimodal inputs. A finite-state representation has distinct advantages over previous approaches, mainly, that it is significantly more efficient, permits "tight-coupling" of multimodal understanding with speech recognition, and provides a more generic framework for multimodal disambiguation.

The principle idea behind a finite-state device is the ability to transition between states based on a current state and a current input. Based on the conditions of the current state, the input would dictate the next state to transition to. A finite-state transducer (FST) is one instantiation of such, with the distinction that an output is generated at every transition between states. That is, an FST is similar to a 2-tape finite state automata, with one designated for input and the other output. Before the rise of multimodal systems, finite-state models have always been deemed a valued mechanism for language processing as it offered, through calculus, elegant solution to integrate constraints from different levels of language processing. The success of finite-state devices in language processing, in turn, motivated the inclusion of finite-state models in a multimodal system.

The finite-state approach for multimodal integration may be discussed in two parts. We will first discuss the parsing mechanism followed by a discussion of the representation of finite-state automata through a finite-state transducer.

Parsing in a multimodal system with n modes requires a finite-state device with n+1 tapes; one tape for each input mode, and one tape to output the combined meaning. A multimodal context-free grammar, a grammar that does not rely on previously discussed material, can be modeled by such an FSA such that each grammatical entity contains n+1 components. For example, in a system that accepts speech and pen inputs, a finite-state device with three tapes is required: one for speech, one for pen-gesture, and a third for the composite meaning. A compatible multimodal grammar for the above system could be one that comprises a speech component, a pen-gesture component, and a composite semantic meaning component for each grammatical entity identified.

Consider the example command, “call <pen-gesture indicating a person> this person.” The grammar would generate the output at the terminals

Terminal → W: G: M

V → call: ε: call

DET → this: ε: ε

S → person: G_p : person

Epsilon is a placeholder for when the component is not found in a particular stream and symbols in W are words from speech stream. On the other hand, the G_i’s represent the specific reference indicated by the pen-gesture, where the subscripts describe the *type* of entity referenced. That is, a particular interface may support three types of gesture references, G_p, G_l, and G_o signifying a pen-gesture reference (from display) to a person, location, and organization, respectively. Similar to the typed-feature structures found in unification, type-categorization in finite-state models adds constraints to the overall semantic interpretation. However, type-categorization presents a more generic framework. In terms of mutual disambiguation, type-categorization provides a better resolution when the speech “is underspecified with respect to semantic type” (p. 3). In other words, if the user said, “ tell me about *this one*” in conjunction with a gesture input of type person, the type-categorization would resolve the ambiguous “this one” to “person.” Moreover, type-categorization also supports the development of a type-hierarchy reflecting the ontology in the system’s domain. For instance, the interface previously mentioned supports a generic gesture type G with subtypes G_p, G_l, and G_o; and if convenient, further specification of G_p can be made for male and female by introducing G_{pm} and G_{pf}.

While an n+1 tape finite-state automaton is theoretically possible, prevalent finite-state language processing tools only support finite-state transducers. The difference between them, as previously discussed before, is that a finite-state transducer (FST) would always generate an output on a separate tape during each transition. In reality, implementing a multimodal architecture using FST remains basically the same. An n+1 tape FSA can be converted into a FST by simple mapping functions between the set of input modes and the final output. In the speech and pen-gesture example, the abstraction function would be τ: (G x W) $\rightarrow$ M, where the transition outputted by both input modes are combined into an input component (G x W) and the output component is M. In other words, the FST model in this system would take in words and pen gestures on the input tape, and the combined meaning on the output tape.

With respect to temporal constraints, the finite-state integration approach presents another simplification by focusing only on enforcing temporal ordering of modalities. Recall that, in a multimodal interaction with written and speech-inputs, the written input always precedes the speech . To process multiple integrations during one interaction, unification-based multimodal integration would require time-stamp verification every time a unification operation is performed. Such strict enforcement of temporal linearity can become expensive and potentially problematic, as an empirical study with users must be conducted in *each* application domain before a precise temporal relationship may be determined. Moreover, as timing between modes is relative to the respective modes, the restriction of the specific temporal constraints may have negative effects on the scalability of a multimodal architecture. The relaxation from explicit temporal constraints to temporal ordering is therefore yet another characteristic simplification of this method.

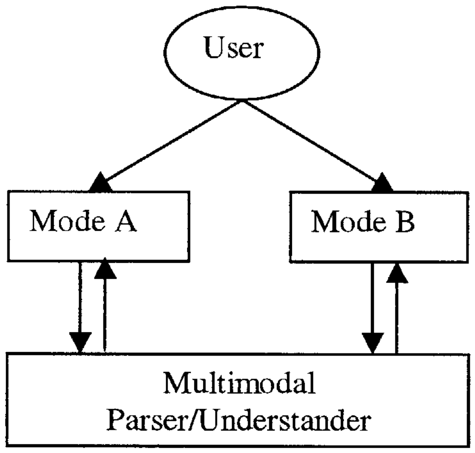

Figure 2 Finite-state transducer based Multimodal Integration Architecture

System: Multimodal Messaging Application

Johnston and Bangalore first implemented the FST-based approach in an application that allowed the user to interact with a company directory through speech and pen. In this system, the user interface consists of an automated speech recognizer and a dynamic screen display that recognizes pen-gesture inputs. Depending on the user's selection, the screen for the system can display different directories, that of companies, of departments, and of people. For example, the user can command, "email this person and that organization" and indicate the corresponding person and organization on screen.

Since the pen-recognition display is dynamic and specific, that the user is either looking at a directory of people, department, or organization, there is little ambiguity in pen-recognition. To best use this property towards disambiguating input streams, this system uses the gesture recognition to dynamically modify a language model. In order to do so, Johnston first uses an FST, FST_1, to map the gesture stream to all the possible word sequences that can co-occur with the observed gestures; abstraction function R, R: G→M. FST_1 is then used to modify the language model. Then, by composing FST_1 with the speech stream using the modified language model, a new transducer, FST_2, containing information from both speech stream and gesture stream is outputted. The last step, then, is to feed the output from FST_2 into a final transducer, RESULT, to capture the abstraction function τ. We now demonstrate with an example: consider this input with a speech string, "D him and B her," in conjunction with 2 pointing gestures, first one indicating man and the later one a woman . Assume, for simplicity, that the system accepts 2 command, to "D" and to "P." Furthermore, assume that the system can "D" a man but not a woman, but can "P" both a man and a woman. FST_1 then would map to the set {D [and] P}. This mapping, in turn, is used to modify the language model such that a legal command can be resolved; instead of "B," the language model would be adjusted to be more receptive of "P." FST_2, in this case, might become {D [and] P}. Finally, combining FST_2 into RESULT would yield {D G_{man} [and] P G_{woman} }, translating into a semantic interpretation of, "D [the man indicated by G_{man}] and P [the woman indicated by G_{woman}]."

The preliminary speech recognition experiments have yielded encouraging results. On a corpus of 1000 utterances, sentence-level error was reduced by 23%.

Chapter 3

Current System: MACK

We now examine multimodal integration technique in the context of kiosk with a paper map. We first offer an overview of the kiosk application, MACK, followed by a detailed problem statement.

3.1 BACKGROUND INFORMATION ON DOMAIN: MACK

Media lab Autonomous Conversational Agent (MACK) is an embodied conversational kiosk for the MIT Media Lab. In the backend, MACK has a database containing relevant information on the consortia, groups, rooms, and demos in the lab. Due to the inherent differences between the entities, information contained in the database is not completely symmetrical. For example, while groups and demos require a physical area, consortia do not. The fact that an open-house schedule is only applicable to demos and no other entities also exemplifies the inherent asymmetry in information. In the front end, MACK interfaces the user with a graphical output of an embodied agent (a life-sized blue robot), a microphone, and a set of paper map and pen. The graphical display is to convey the gestures that are synchronized with the speech outputted to MACK. The microphone is to receive user's verbal utterances, while the map and pen to sense pen-gestures. This design is aimed to create an interactive and natural interface using natural language abilities and shared physical artifacts [2, 3].

3.2 PROBLEM STATEMENT: how is this problem interesting or different?

To design a multimodal integration architecture best for the domain of an embodied kiosk with the shared "map."

The main idea here is to create an effective multimodal input integration algorithm for kiosk involving speech and a paper map. As seen from above, there have been several systems that have used both speech and pen-gesture recognition as input modalities. However, the kiosk application is unique in many aspects. Fundamentally, the interaction between a kiosk and a user is different than any command-based systems. By assumption, the user of a kiosk system should have special interest in the information that the kiosk has to provide. In fact, it may be further stated that the user of a kiosk system does *not* know the correct response to anticipate, and is *not* always correct. We differentiate between a command-based system and a kiosk system by the expectation on and the expertise of the user. A user of the command-based system is sufficiently knowledgeable of the information built in the application, while that of the kiosk more than likely does not. For instance, a Jeanie user would *tell* his/her talking calendar to cancel a meeting and does not need the confirmation from his/her talking calendar to determine what meeting to go to next. Similarly, a QuickSet trainee may rely on the system to simulate the consequence of his/her last action, but would not completely depend on QuickSet's output to determine where to place the next platoon; instead, the user is likely to have sufficient information on the simulation strategies such that the output from QuickSet would merely be advisory.

This assumption, of user's insufficient knowledge, bears significant consequences. First, that the kiosk has an enormous responsibility to provide accurate information to the users' queries. While the importance of performance accuracy is traditionally strived for, it is of particular emphasis here because the user may not distinguish between correct and incorrect responses. That is, if the user were to query for directions to group A and received the directions group B, then the user would be misled without knowing. In contrast, in setting up a scenario for simulation, a QuickSet user does not need information from the system to proceed to the next action; the user will likely have a simulation strategy in mind that is independent of what the map will display, where the map merely serves as a graphical tool that allows for distributed information sharing [9]. Moreover, the accuracy of the system is also intimately related to the users' frustration level. That is, unless the system accurately understands the user, any information that it provides would only confuse the user more. Therefore, the kiosk system should always prioritize in understand the user's queries.

3.2.1 How is this different than all the previous multimodal systems?

In practice, this affects the integration of different input devices into the system. In Johnston's multimodal messaging system, the pen-gestures are involved in a feedback loop during an optimization to find the best fitting finite-state transducer, and a most probable solution is always applied [11]. This approach may not be appropriate in the kiosk system, since the "best" is not necessarily content-correct [15]. As an example, consider the scenario where the user may point to group A on the map and requests for information on an unknown group X. Ideally, the kiosk should be able to detect the confusion, if not query the user for clarification between group A and group X. Using Johnston's deterministic optimization algorithm, integrating the two modalities may result in a best guess that the user is requesting about group A. If the user were inquiring about group A, then this best guess is indeed correct, and the output to the user would benefit the user. However, in the case that the user had inquired about group X, then the output from Johnston's integration algorithm would only add to the users' confusion. As previously mentioned, avoiding confusion should be made one of the top priorities in the system.

The second distinction in designing the integration algorithm for a kiosk is the basic informational structure that a kiosk embodies, in combination with the shared space that the system intends to preserve. In this system, the shared reality between the user and the kiosk is the paper map. The user is able to indicate particular coordinates to the kiosk, as well as making notes for himself/herself. This creates a great differentiation from previous tasks, as the paper map inherits ambiguities commonly seen in human-to-human interactions. In particular, the map used in MACK contains asymmetrical information, that is, certain spatial locations have associated room numbers while others have associated room names, or that certain rooms are occupied by groups while others are not. The non-uniform information represented by this paper map constitutes a remarkable distinction from other shared-reality maps; the map in QuickSet is a dynamic one as the system permits the user to change perspectives and views using commands such as “zoom in,” “scroll here,” “pan,” and “show me the hospital [on the map]” [17]. Moreover, the map used in QuickSet embodies a standardized one-level hierarchy for the spatial locations displayed---each location either has one or zero unique property associated with it [10].

At a mere glance, this does not seem much. But consider the properties that are associated with the above listed attributes; for example, consider the various projects associated with each group that would be demonstrated in the respective rooms, then consider the matter of available show-times associated with each demonstration. The use of a physical map undoubtedly enhances the naturalness of the interaction, as it provides the user a more familiar channel of communication. Moreover, it is likely a more concise representation to the user, with neither redundancy of information nor cluttering of useless information. For example, a room has a name label only when it has a valid name distinct from the room number, and rooms only have group labels if there are groups occupying them. In practice, this avoids the embarrassment of having to label conference rooms and restrooms in the building with “no groups are here.” Moreover, the kiosk is designed such that the hierarchy between groups and projects is assumed, just as the relationship of an artist and his work can be assumed. Consequently, like that if a map of an art museum would need only to label an exhibition room with the name of the artists whose work is being shown, the map used here shows only the names of the groups and not the individual projects. The embedded knowledge here is remarkable for it exemplifies the need of an algorithm by which to obtain information at appropriate levels of granularity. Unlike systems that incorporate a dynamic map to allow users to manually scroll and zoom, thus permitting the user to "self-structure the map display" such that the entities indicated would be directly "tailored to the user and task," [15] this kiosk here must embody its own disambiguation algorithm to tailor to the user and the task.

3.3 Design Overview

Having identified the unique context of an embodied conversational kiosk, we now identify issues considered during the design of the multimodal integration algorithm for a kiosk system.

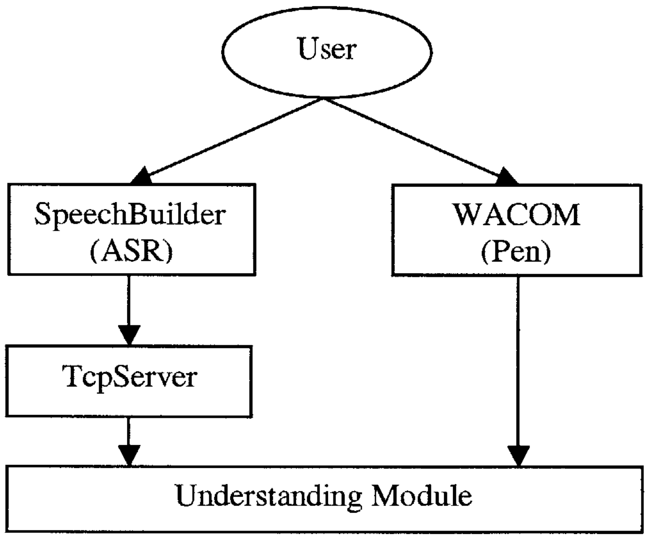

Figure 3 MACK Multimodal Architecture

3.3.1 Design Issues:

As discussed before, the top priority of the integrator is accuracy. In the general context of multimodal architectures, this implies choosing a framework that has the best error handling, and preferably, one that would maximize mutual disambiguation (MD) between the input modes. Moreover, since we have distinguished the kiosk domain as a unique class among multimodal systems, this multimodal integrator should be designed to minimize any loss of generality. That is, we should find an approach that is fit for MACK as well as for an entire class of kiosk systems. From previous studies, we have seen that speech-driven systems such as Put-That-There and CUBRICON use little MD during its integration, since resolution of deictic terms would account for less than half of the anticipated multimodal interactions. In keeping with preserving generality, the improvement to typed feature structures from "slots" suggests that the unification-based approach is preferred to the frame merging technique.

Both unification-based and the finite-state transducer (FST) approaches have shown great strengths in terms of MD. However, as previously mentioned, the expressiveness of unification-approach comes at a high price. In general, the combinatorial complexity of the unification-approach makes the integration technique unscaleable, which is inconsistent with our objective to be as generic as possible. One source of this complexity in the unification-based framework, is the need to enforce specific temporal constraints. There is no doubt that enforcing specific temporal constraints would provide greater multimodal resolution than if only temporal ordering was preserved. However, it is not entirely clear if the added marginal benefit would justify the increase complexity. For example, consider the scenario where the user makes the query, "what is this and that" while using the pen to indicate group A and group B. Enforcing strict temporal constraints would compare the time stamps of pen-indication and deictic terms, and conclude that "this" corresponds to group A and "that" corresponds to group B. Preserving only temporal ordering would render the same resolution without having a priori knowledge of temporal correlation between the input modes.

In application, the relaxation from strict temporal assumptions is a great simplification. Consider the implications of having to disambiguate multiple input streams using time stamps. With respect to an automated speech recognition (ASR) tool, the approach would require that the device provide time stamps at the phoneme level. Unfortunately, this requirement presents a severe limitation on implementation, as most available ASR's currently do not support phoneme-by-phoneme time stamping. Although there are heuristics for approximating these time stamps, most of those heuristics are often based on the assumption that individual words are equal length; phoneme time stamps are computed by dividing total utterance length by the total number of words in the utterance. Since the lag time between gesture and spoken deictic is only 1.4 seconds, the gross approximation using averages may easily introduce large error margins [16].

Another important aspect to inspect is the inherent ambiguity in the interface of this kiosk system. Records of interactions between the semi-autonomous MACK (operated by Wizard-of-Oz) have shown repetition of utterances among queries. Users seem to infer different classes of queries, such as a general description or directions to an entity, and use similar (if not the same) sentence structure to generate queries within the same class regardless of the entity in question. For example, users are observed to reuse the sentence structure, “tell me about X” while replacing X freely from the names of consortia, groups, and demos. This thematic repetition suggests an advantage of representation by type-categorization over typed feature structures. Recall that type-categorization can constrain the overall semantic interpretation, where typed-feature structures would simply be unified or filtered. If an unification approach is taken, the use of typed-feature structures in this context would not be able to offer MD. A new probability tree would be searched in exploration of the most probable type for X. In contrast, the FST-model would exploit the type-categorization to constrain the number of possible semantic interpretations.

Another way of saying that type-categorization constrains semantic interpretation is that it effectively detects conflicts. As we have emphasized before, we prefer confirmation with user to generating response based on a guess. The type-categorization, therefore, allows elegant detection of conflicts. Although the decision to actually generate a confirmation to the user is beyond the scope of multimodal integration, at the kiosk-system level (which considers multimodal input, multimodal integration, and multimodal output), conflict-detection can be used to as a signal for the system to respond with a query for user clarification. Moreover, using FST, the transitional outputs generated prior to the conflict detection would be able to offer sufficient context to generate, as an output, a specific query for the user. Consider the following query: the user says, “tell me about Group_A” while pointing to Group_B, an integration framework that embodies conflict-detection would detect the inconsistency and have specific knowledge that the conflict is between "Group_A" and "Group_B" in generating the system's understanding. This information, in turn, may be used generate the following confirmation, "would you like to know about Group_A or Group_B?" Without the transitional outputs, the system would not have sufficient information to request user's confirmation. Taking one step further, without a conflict detection mechanism, the system would not be able to determine when a confirmation is needed.

3.3.2 Applying Multimodal Finite-State Transducers

There are different methods of integrating finite-state transducers (FST) with speech and pen recognition. As Johnson puts it, the "best approach to take depends on the properties of the particular interface to be supported" [p3, 11]. Certainly, we know that the final abstraction function would be one that maps speech and pen input to a semantic interpretation output, $\tau^*: (P \times S) \rightarrow M$. The task now is to design the intermediate steps such that MD is maximized without lost of generality.

As mentioned before, one distinguishing feature of the conversational kiosk with shared space, is the asymmetry presented by both the interface and in the physical building's information structure. Such lack of uniformity, in information, is useful towards conflict detection in type-categorized analysis. Recall that, in Johnston's messaging application [11], transitional transducers were created before a final transducer mapping of the function $\tau: (G \times W) \rightarrow M$ could be generated. Recall also, that it was through these transducers that the space of semantic interpretation is narrowed. For example, the transducer representing $G \rightarrow W$ would constrain the speech inputs to those that are compatible with the gesture stream. For our kiosk system, we can take a similar approach: using a less ambiguous input stream to constrain the more ambiguous. Given that only groups and demos occupy a physical space and that consortia do not, a pen-detection on the map would refute the validity of questions about any consortium. Consequently, if an ambiguous object type, such as "this" or "it," is recognized in speech, the G $\rightarrow$ W transducer would omit consortium as a valid object type. Conversely, if an object is recognized from the pen-input (a pen-input can indicate a room or a group) while a particular room name is recognized in the speech input, then the W$\rightarrow$G transducer would constrain the object type to be room.

We propose using a transducer to capture the relationship between pen-input and speech. Knowing, for example, that a pen-input is constrained by what is physically indicated on the map, we can conclude that a pen input can be only of certain types, namely groups and rooms. The ambiguity in the physical map would prevent us from drawing further conclusions about the specific type. But even so, this is sufficient constraint for a transducer that captures the relationship between a pen-input and speech input. That is, when a pen input is received, we should verify that the object in the speech query is of type *Room* or *Group*. Otherwise, a conflict should detected. Conversely, as the pen-input is not specific, we would also need to verify that the object type in the speech query is *exclusively* either Room or Group, or a conflict should be detected. This allows for speech to disambiguate pen-input. Furthermore, as there is a one-to-one mapping between pen-input and speech-input on particular object names, the transducer should also verify the precision of the object instances. Given such, we are now ready to define a transducer *Pen_Speech* that captures the relationship R: P$\rightarrow$ S.

Moreover, having known that different entities in the building are associated with different properties, we can create a second transducer, *Type_Speech* which represents the mapping (P x S_t) → S. In *Type_Speech*, we use the type of the object to constrain the type of queries acceptable. For example, having known that only groups and demos occupy a physical space, the detection of an object of type *Consortium* would signal conflict.

Let us take one-step back and examine how all the transducers would work together during integration. The first step is for the individual input devices to process the incoming streams, then construct three finite state machines *Pen*, *Speech*, and *SpeechType* to represent the range of pen-inputs, the sequence of spoken inputs, and the set of spoken input that corresponds to particular types, respectively. We first compose *Pen* and *SpeechType* with *Type_Speech* to generate the transducer *Type_Lang* representing the relationship between the set of input sequences and all the possible word sequences that can co-occur. We then compose *Pen* and *Pen_Speech* separately to generate a transducer *Pen_Lang* that captures the constraint between the particular sequence of pen-gestures in *Pen* and the word sequence. Finally, we consolidate *Speech* with each *Type_Lang* and *Pen_Lang* to check for discrepancies and conflicts. If no conflicts are found, then we compose *Pen* and *Speech* with τ^*: (P x S) → M. This would yield our final transducer that outputs the resolved meaning.

As we previously discussed, there is a variety of different conflicts that can be detected. Each type of conflict is represented by a different value, which would decide the type of confirmation that the system would output to the user. In addition, during the course of composing these transducers, the transitional output values would provide the specific *context* of the conflict. From the above discussion, we may conclude with following different types of conflicts:

1. speech inconsistency: detected in *Type_Speech* when the object type is not compatible with query. I.e. "where is *Consortia_A*"
2. speech insufficiency: detected in *Type_Speech* when no object instance is specified. I.e. "what is this" with no pen indication of a group or room.
3. speech-pen inconsistency: detected in ***Pen_Speech***, *Type_Speech*, *Type_Lang*, or ***Pen_Lang*** when either the object type or the specific object-name is inconsistent between the pen- and speech-input. I.e. "what is Group_A" with pen indication of Group_B.
4. pen insufficiency: detected in ***Pen_Lang***, when the pen-input is not disambiguated by speech. I.e. pen indication with no speech input.

When a conflict is detected, a confirmation is generated based on the transitional output from composing the transducers. Let us clarify using examples. Consider the following inputs to MACK:

1. user says "where is Consortium_A" in conjunction with pen indication of Group_A.
2. user says "what is Group_A" in conjunction with pen indication of Group_A.

In case 1, *Pen* ={Group_A}, *Speech* = {where, is, Consortium_A}, *SpeechType* = {Consortium}. Composing *Pen* with *Type_Speech* would be trivial, yielding feasible set of {where is group, what is group }. Composing *SpeechType* with *Type_Speech*, on the other hand, would flag conflict of type 1, since no consortium occupies a physical space.

Therefore, no acceptable transducer *Type_Lang* can be generated. Similarly, when we compose *Pen* and *Pen_Speech*, we would also encounter a conflict of type 3, as Group_A is not the same as Consortium_A. Consequently, no acceptable transducer *Pen_Lang* can be generated. Since conflict is detected, we cannot consolidated *Speech* with either *Type_Lang* or *Pen_Lang*. On the other hand, the transitional outputs from composing the transducers provide the contexts, "where is," "Consortium_A," and "Group_A." Using this information, the system can then generate the following confirmation[4], "There are no spaces dedicated to consortia. Would you like information on Consortium_A?"

In case 2, *Pen* ={Group_A}, *Speech* = {what, is, Group_A}, *SpeechType* = {Group}. Composing *Pen* with *Type_Speech* would be trivial, yielding feasible set of {where is group, what is group}. Composing *SpeechType* with *Type_Speech*, would also yield the same feasible set. Therefore, *Type_Lang* is generated to accept {where is group, what is group }. Similarly, when we compose *Pen* and *Pen_Speech*, we would obtain {Group_A} as the transducer *Pen_Lang*. Finally, we can consolidated *Speech* with *Type_Lang* or *Pen_Lang* to generate the final meaning, {what is Group_A}.

[4] The decision on the type of confirmation to be made is arbitrary. The example demonstrates a confirmation based on answering to conflict type 1. A possible confirmation in response to conflict type 3 may be, "Would you like information on Consortium_A or Group_A?"

3.4 Implementation

In between the sensory input devices and the output generation tools, MACK is implemented completely using Java JDK1.2.2.

SpeechBuilder version 1.0 is the automated speech recognition (ASR) device of choice for this implementation. It was selected for its ease of use and better-than-average performance compared to commercial universal-user ASR. SpeechBuilder is a development tool produced by the Spoken Language Systems Group at the MIT Laboratory for Computer Science. SpeechBuilder has an embedded application program that allows the user to specify semantic interpretations based from the language understanding component. But for our purposes, only the basic speech recognition functionality will be used. Since we will define a multimodal grammar, we only anticipate a processed language stream, such as a string of words as uttered by an user, as the output of SpeechBuilder.

To receive pen-gesture inputs, a Wacom Graphire pen and tablet set is used. The Graphire set contains a pressure-sensitive pen used to track actions on the tablet. By embedding the tablet in a position directly underneath the designated map location, the tool indirectly allows the system to track user's pen movements across the maps. The pen-gesture recognition is implemented using Java Application Windows Toolkit (AWT) package to recognize and hash single-point coordinate values on the table to applicable object values. As soon as a pen-indication is sensed, an object value is stored in a vector. This vector of such values thus comprises the pen-gesture input of our system.

A limited context-free multimodal grammar is defined here. For our purposes, three demonstrative sentence structures are defined in the grammar. Namely, the accepted speech structures are as follows: (1) "where is X (and Y);" (2) "what is X (and Y);" and (3)"when can I see X (and Y)." We feel that these three questions compose a representative set of multimodal integration using the transducers mentioned above. That is, these three structures would demonstrate the effectiveness of the FST-approach to detect conflicts. For example, we should see that the query, "where is Consortium_A" signal conflict, as predicted in *Type_Speech.* Similarly, the query "what is <pen input> Demo_A" should also signal conflict, as dictated from *Pen_Speech.* See Table 1 and 2 for the list of conflicts predicted by the transducers.

The multimodal grammar works as follows:

When a pen-gesture is recognized by the Wacom tablet, the recognized object is sent to the UM and stored in a vector P. When an utterance is recognized by SpeechBuilder, the text of the speech is sent to UM, where the text is tokenized into words and stored in vector S.

After a complete utterance has been received and tokenized in the UM, the parsing process is initialized. The multimodal grammar takes as argument, S and P. Multimodal parsing is done by iterating through S and P for inputs, where each iteration terminates when the inputs in S have been enumerated. Each state in a transducer is uniquely represented by an instance of a state object. Each new parse begins in the state "init" and uses an input from each S and P to transition to the next state. During each transition, values from P and S are checked against values in the dictionary and the inconsistency matrix. This process of checking values is functionally equivalent to running *Type_Lang* and *Pen_Lang*. The value generated from verification in the dictionary the inconsistency matrix is then written to a temporary vector, *CurrentQuery*. The parse concludes when it has exhausted the inputs in *Speech.* At this point S and P are reset to empty vectors.

At the end of the multimodal parse, *CurrentQuery* would have valid fields to generate a semantic interpretation. *CurrentQuery* would have at least 5 fields, *conflict_type*, *speech_object_type*, *pen_object_type*, *speech_object_name*, and *pen_object_name*; *conflict_type* has denotes the type of conflict detected in multimodal parse, that is, 0 if no conflict is detected, and a unique value for each of the above mentioned conflict types. The later two arguments, **_object_name* , comprise the arguments that would become either the object of a specific query (when no conflict is detected), or as the context of a conflict. For example, if, at the end of multimodal parse, *CurrentQuery* has 3, Group_A, Group_B, for *conflict_type*, *speech_object_name*, *pen_object_name*, respectively, then a confirmation requesting the clairification between Group_A and Group_B may be generated.

The semantic interpretation is written to *MultimodalFST*, and *CurrentQuery* is reset to empty. Building up with the number of queries the user submits, *MultimodalFST* effectively maintains a thorough context history of the interaction.

It is necessary to reset P to an empty vector at the end of every parse because we are not supporting timing constraints. The boundary-setting heuristic used here is intuitive: include all pen-input between the end of the previous utterance and the end of the current utterance. This heuristic is consistent with preserving temporal ordering without temporal constraints. The correctness of this algorithm can be proven inductively.

3.4.1 Code Modules and Methods

Dictionary

This is where the vocabulary of the grammar is defined. In addition, this is also the class that assigns the type category for objects in the system. There are 5 categories of objects here, namely, consortium, deictic term, group, demo, and room. Each of the categories contains a subset of names from the comprehensive database of the original MACK.

TcpServer

This is where the input from ASR is received and where a finite-state machine for the speech input is constructed.

Understanding Module (UM)

Abstraction Functions:

1. Representation of finite-state machines:

   Pen and *Speech* are represented by vectors P and S, respectively.

2. Representation of finite-state transducers (FST):

   Pen_Type and *Pen_Speech* are combined and summarized into an inconsistency matrix that constrains the relationship between input object types and semantic interpretation. A matrix is an appropriate selection because the relationships represented by these two FST's are not dynamic. We know, *a priori*, the constraints between the inputs due to the asymmetrical information structure in the system.

Type_Lang and *Pen_Lang* are not explicitly represented here. The role of these two FST's is to verify consistency between the input streams. Conversely, we argue that the output of running these two FST's can be extracted by searching for output values from the inconsistency matrix. We argue here that a concrete representation of these FST's is not necessary.

τ^* is represented by a finite state machine. The finite state machine is moderated by a multimodal grammar. The inputs to τ^* are represented by the vectors *Pen* and *Speech* and the output by another vector, *MultimodalFST.*

At the first glance, it may seem like poor engineering to implement inconsistent representation. However, we argue that a consistent representation is not cost-effective. For instance, if a unique finite state machine were used for each FST, the system would mandate multiple iterations of the same input set. However, this repetition preserves representational consistency at an unnecessary cost of performance efficiency.

<u>Methods</u>

multiParse

arguments: two vectors representing the input streams from speech and pen

modifies: appends the final semantic interpretation from this parse to the vector that contains the final semantic interpretation.

returns: a vector containing values that map to a semantic interpretation *multiResolve*

arguments: a vector containing values that map to a semantic interpretation

modifies: nothing

returns: a proper response to the pertinent interpretation. That is, if no conflicts are observed, the information queried by the user is returned. Otherwise, a specific query for clarification is generated for the user.

writeToTape

arguments: none

modifies: nothing

returns: a vector containing values that map to a semantic interpretation pertaining to the most recent query

<u>Class</u>

States:

{init, when, when_can, when_can_I, when_can I, when_can I_see, when_can I_see_pen, when_can_I_see_X, when_can_I_see_pen_X, what, what_is, what_is_pen, what_is_X, what_is_pen_X, where, where_is, where_is_pen, where_is_X, where_is_pen_X, queryUser}

Chapter 4

Performance Evaluation

Although the current multimodal implementation has not undergone user testing, MACK has been tested with a sample of ten of speech and pen interactions. See Table 3 for details on the output

Table 3. Test data

	Interaction	**Semantic Interpretation**	**Response**	**Remark**
1.	"where is Consortium_A"	No location associated with type consortium *Conflict*	"Consortia do not have a physical location"	TP
2.	"where is <pen@A> Demo_A"	Map cannot indicate type demo *Conflict*	Query user if Group_ A or Demo_A were of interest	TP
3.	"what is <pen@B> this and Group_A"	<pen@B> = Group_B "this" = Group_B *Query*: information on group B and A	Description of Group_B and Group_A	TN
4.	"where is Group_A and <pen@B > this "	<pen@B> = Group_B "Group_ A" = Group_B "this" = empty *Conflict*	Query user to specify which group	FP

5.	"what is <pen@A> <pen@B> this"	<pen@A> = Group_A "this" = Group_A *Query*: information on Group A	Description of Group_A	FN
6.	"when can I see Demo_A and Demo_B"	*Query*: information on Demo_A and Demo_B	Description of Demo_A and Demo_B	TN
7.	"when can I see <pen@A> this and Demo_B"	<pen@A> = Room_A "this" = Room_A *Query*: open house information on Room_A and Demo_B	Open house times for Room_A and Demo_B	TN
8.	"when can I see Group_A"	No open house information associated with type group *Conflict*	"Open House times are associated with specific demos, or particular rooms"	TP
9.	"when can I see <pen@A> this"	<pen@A> = Room_A "this" = <Room_A> *Query*: open house information on Room_A	Open house times for Room_A	TN

Legend:

TP: True positive: conflict identified when user input is ambiguous
FP: False positive conflict identified when user input is unambiguous
FN: False negative: conflict not identified even though user input is ambiguous
TN: True negative: conflict identified and user input is clear.

While the results from this set of sample outputs would not represent accuracy in practice (we will need user testing on the system to determine that), the set of false positives and false negatives are indicative of the implementation limitation. In particular, interaction 5 exemplifies the class of problems in matching multiple objects. This error may be attributable to lack of timing constraints to moderate the integration of multiple speech and pen-input objects.

Nevertheless, we argue that the FST-framework is an effective one. The implementation demonstrated that type-categorization is indeed advantageous for interactive systems with unrestricted semantic types in input modalities by effectively incorporating mutual disambiguation. Recall that while the paper map facilitated more naturalness in the interaction, it lacked standardization in the representation of information. As more and more systems strive to facilitate naturalness and/or to establish shared-space, individual input modalities are likely to absorb more ambiguity. Consequently, categorizing semantic meanings would only become more difficult. Using type-categorization, the FST-framework offers a promising potential for future multimodal systems.

Chapter 5

Future Work

Preliminary test data suggests that an appropriate extension of the integration algorithm is the inclusion of multi-object resolution techniques. The current parser preserves temporal ordering for inputs from each modality. The system has no further information regarding the overall temporal linearity of both input modes. In the case of interaction 5, the parser erroneously matches the first acceptable type-object from the pen input to that from the speech input, therefore, "Group_A" from speech is matched to "Group_B" from the map input. More specifically, note that temporal ordering is insufficient for instances where the number of expected objects may be different between the different input modes; in interaction 5, "where is Group_A and <pen@B> this," speech input provided 2 objects ("Group_A" and "this") while pen input provided only 1 (B). This inconsistency in number of objects across the modalities, in turn, can be attributed to the acceptance of heterogeneous reference types in speech-input; namely, proper names and deictic. When a proper name is used in speech, the user may be ambivalent to accompany the query with a pen-referent, where as he/she is more likely to provide one when a deictic term is used. A direct solution would be to incorporate time stamps, and compare active time windows of objects before integration.

An alternative approach to the multiple object problem, may be to implement a rule-based system that assigns integration tags to objects based on type. For example, the conflict in interaction 5 may be avoided if we had chosen to integrate the pen-input with the deictic term instead of the other object. Hence, we may specify a rule to prioritize integration for objects of type deictic. In other words, if the number of typed objects from speech is different from that from the pen, always integrate the deictic object with a pen-input object before integrating other terms.

Another possible extension to MACK, would be to increase the practical application of the system. The current system is implemented with limited vocabulary and grammar for demonstrative purposes only. The limited grammar can be replaced by incorporating a pre-defined, extensive grammar.

Chapter 6

Conclusions

A multimodal integration module is designed and implemented for **M**edia Lab **A**utonomous **C**onversational **K**iosk (MACK). We have examined several powerful multimodal integration frameworks, and have designed MACK's integrator based on the technique which would maximize mutual disambiguation of speech and pen-input. The final integrator is built based on Johnston's finite-state transducer (FST) approach. Considering the context of MACK, we felt that the FST approach offered the most simple, cost-effective, and scaleable solution.

References

1. R.A. Bolt, "Put that there: Voice and gesture at the graphics interface," *Computer Graphics*, vol. 14, no.3, pp. 262-270, 1980.
2. Cassell, J. "Nudge Nudge Wink Wink: Elements of Face-to-Face Conversation for Embodied Conversational Agents", in Cassell, J. et al. (eds.), *Embodied Conversational Agents*. MIT Press, Cambridge, MA, 1999.
3. Cassell, J., Ananny, M., Basu, A., Bickmore, T., Chong, P., Mellis, D., Ryokai, K., Smith, J., Vilhjálmsson, H., and Yan, H., Shared Reality: Physical Collaboration with a Virtual Peer. *Proc. CHI 2000*, pp. 259-260, The Hague, The Netherlands, 2000.
4. P. R. Cohen, M, Dalrymaple, D. B. Moran, F. C. N. Perreira, J. W. Sullivan, R. A. Gargan, J. L. Schlossberg, and S. W. Tyler, "Synergistic use of direct manipulation and natural language, " in *Human Factors in computing systems: CHI'89 Conference Proceddings.* New York: Addison-Wesley, pp. 227-234, Apr. 1989; reprinted in *Readings in Intelligent User Interfaces*, M. Maybury and W. Wahlster, Eds. San Francisco, CA: Morgan Kaufman, 1998.
5. P. Cohen, M. Johnston, D. McGee, S. Oviatt, J. Pittman, and I. Smith, L. Chen, and J. Clow, "Quickset: Multimodal interaction for distributed applications, " in *Proc. Fifth ACM Int. Multimedia Conf.* , pp. 31-40, New York, 1997.
6. Dale, Robert. *Generating Referring Expressions.* The MIT Press, 1992.

7. Fromkin & Rodman. "Chapter 6: Semantics". *An introduction to linguistics*, 4th edition, 1983.

8. S. C. Hirtle, M. E. Sorrows. "Designing a Multi-modal Tool for Locating Buildings on a College Campus." *Journal of Environtmental Psychology*, vol. 18, pp. 265-276, 1998.

9. M. Johnston, P. Cohen, D. McGee, S. Oviatt, J. Pittman, and I. Smith, "Unification-based multimodal integration, " in *Proc. 35th Annu. Meeting Association for Computational Linguistics*, San Francisco, CA, 1997, pp 281-288.

10. M. Johnston, "Unification-based multimodal parsing, " in *Proceedings of COLING-ACL*, Montreal, Canada, 1998b, pp 624-603.

11. M. Johnston, S. Bangalore. "Finite-state multimodal parsing and understanding," *Proceedings of COLING-2000*, Saarbruecken, Germany. 2000.

12. M. Johnston. "Deixis and conjunction in multimodal systems," *Proceedings of COLING-2000*, Saarbruecken, Germany. 2000.

13. D. R. McGee, P. R. Cohen, "Creating tangible interfaces by augmenting physical objects with multimodal language," *IUI'01*, Sante Fe, New Mexico, January 14-17, 2001.

14. J. G. Neal and S. C. Shapiro, "Intelligent multimedia interface technology, " in *Intelligent User Interfaces*, pp. 11-43, J. Sullivan and S. Tyler, Eds. New York: ACM , 1991.

15. S. Oviatt, "Multimodal interfaces for dynamic interactive maps, " in *Proc. Conf. Human Factors in Computing Systems: CHI'96*, pp 95-102, Vancouver, B.C., Canada, 1996.

16. S. Oviatt, A. DeAngeli, and K. Kuhn, "Integration and synchronization of input modes during multimodal human-computer interaction, " in *Proc. Conf. Human Factors in Computer Systems: CHI'97*, pp 415-422, 1997.

17. S. Oviatt, "Mutual disambiguation of recognition errors in a multimodal architecture, " in *Proc, Conf. Human Factors in Computing Systems: CHI'99*, pp. 576-583, Pittsburgh, PA, 1999.

18. S. Oviatt, "Ten myths of multimodal interaction," *Communications of the ACM*, vol. 42, no. 11, pp. 74-81, 1999.

19. S. Oviatt, P. Cohen, "Multimodal interfaces that process: What comes naturally," *Communication of the ACM*, vol. 43, no. 3, pp 45-53, 2000.

20. Thórisson, K. R.. Computational Characteristics of Multimodal Dialogue. *AAAI Fall Symposium on Embodied Language and Action*, Massachusetts Institute of Technology, Cambridge, Massachusetts, 102-108, November 10-12, 1995.

21. Thórisson, K. R.. Real-Time Decision Making in Multimodal Face to Face Communication. *Second ACM International Conference on Autonomous Agents*, Minneapolis, Minnesota, 16-23, May 11-13, 1998.

22. Torres, O., Cassell, J., & Prevost, S.. "Modeling Gaze Behavior as a Function of Discourse Structure." *First International Workshop on Human-Computer Conversations.* Bellagio, Italy, 1997.

23. B.Tversky. "Cognitive maps, cognitive collages, and spatial mental models." In A. U. Frank and I. Campari (Eds), *Spatial information theory: Theaoretical basis for GIS*, Springer-Verlag, Heidelberg-Berlin.

24. B. Tversky, P.U. Lee. "How Space Structures Language." *Spatial Cognition*, pp 157-179, 1998.

25. B. Tversky, P.U. Lee. "Pictorial and Verbal Tools for Conveying Routes." *COSIT*, pp 51-64, 1999.

26. B. Tversky, "Some Ways that Maps and diagrams communicate," *Spatial Cognition II*,LNAI 1849, pp.72-79, 2000.

27. M. T. Vo and C. Wood, "Building an application framework for speech and pen input integration in multimodal learning interfaces, " in *Proc. IEEE Int. conf. Acoustics, Speech, and Signal Processing*, pp. 3545-3548, Atlanta, GA, 1996.

28. L. Wu, S. L. Oviatt, P. R. Cohen, "Multimodal integration---a statistical view," *Proc. Of IEEE Transactions on Multimedia*, vol. 1, no. 4, pp. 334-341, December 1999.