Natural Language Engineering **11** (2): 159–187. © 2005 Cambridge University Press
doi:10.1017/S1351324904003572 Printed in the United Kingdom

Finite-state multimodal integration and understanding

MICHAEL JOHNSTON
AT&T Labs – Research, 180 Park Ave, Florham Park, NJ 07932, USA
e-mail: johnston@research.att.com

SRINIVAS BANGALORE
AT&T Labs – Research, 180 Park Ave, Florham Park, NJ 07932, USA
e-mail: srini@research.att.com

(*Received 4 June 2002; revised 29 October 2003*)

Abstract

Multimodal interfaces are systems that allow input and/or output to be conveyed over multiple channels such as speech, graphics, and gesture. In addition to parsing and understanding separate utterances from different modes such as speech or gesture, multimodal interfaces also need to parse and understand composite multimodal utterances that are distributed over multiple input modes. We present an approach in which multimodal parsing and understanding are achieved using a weighted finite-state device which takes speech and gesture streams as inputs and outputs their joint interpretation. In comparison to previous approaches, this approach is significantly more efficient and provides a more general probabilistic framework for multimodal ambiguity resolution. The approach also enables tight-coupling of multimodal understanding with speech recognition. Since the finite-state approach is more lightweight in computational needs, it can be more readily deployed on a broader range of mobile platforms. We provide speech recognition results that demonstrate compensation effects of exploiting gesture information in a directory assistance and messaging task using a multimodal interface.

1 Introduction

Multimodal interfaces are systems that allow input and/or output to be conveyed over multiple channels such as speech, graphics, and gesture. These interfaces stand to play a critical role in the ongoing migration of human-computer interaction from the desktop to mobile wireless devices such as portable digital assistants, tablet computers, and next generation telephones. These devices offer limited screen real estate making complex graphical user interfaces infeasible. They also lack a keyboard or mouse making speech and pen the primary input modes. Multimodal interfaces are also important for other keyboard-less settings such as public information kiosks and wall-size displays for remote collaboration and presentation.

Multimodal interfaces enable more natural and effective interaction since different kinds of content can be conveyed in the modes to which they are best suited

https://doi.org/10.1017/S1351324904003572 Published online by Cambridge University Press

(Oviatt 1997). Our specific concern here is with multimodal interfaces supporting input by speech, pen, and touch, but the approach we describe has far broader applicability. We will refer to pointing gestures made on a touch screen with a stylus as *pen gestures*.

To realize their full potential, multimodal interfaces need to support not just input in one mode or another, but single multimodal utterances optimally distributed over the available modes (Johnston, Cohen, McGee, Oviatt, Pittman and Smith 1997). Nigay and Coutaz (1993) present a framework for the classification of multimodal interfaces based on the way in which modalities are used, whether they are fused, and the level of abstraction of fusion. Our approach is directed towards *synergistic* interfaces; that is, interfaces that support parallel inputs which are fused to determine the user's intention. An example of a multimodal utterance from the application we describe in this paper would be to point at a person and then an organization on a graphical display and say *"email this person and this department"*.

Research on interactive systems that support multiple input modalities as well as produce output involving multiple modalities has been active since the early eighties (see section 2). More recently, Johnston (1998b; 1998a) has proposed an approach that integrates multiple input modes with the use of unification-based grammars. In this paper, we provide a different approach to the problem of multimodal integration. In contrast to the unification-based approach, which assumes separate grammars for speech understanding and for multimodal integration, our approach relies on a single grammar of multimodal human language. Furthermore, integration of speech and gesture information and multimodal understanding are performed by a single finite-state device (Johnston and Bangalore 2000; Bangalore and Johnston 2000). With certain simplifying assumptions, parsing and understanding with multimodal grammars can be achieved using a weighted Finite-State Automaton (FSA). The automaton can be considered as running on three tapes which represent speech input (words), pen gesture input (gesture symbols and reference markers), and their combined interpretation. The FSA allows for the modes to interact and constrain each other directly.

The layout of the paper is as follows. In section 2, we survey the existing literature on multimodal systems. In section 3, we describe a multimodal directory assistance application that we used as the context for this research. Section 4 provides background on finite-state language processing. In section 5, we define and exemplify Multimodal Context-Free Grammars (MCFGs) and their approximation as multimodal FSAs. We describe our approach to finite-state representation of meaning and explain how the three-tape finite state automaton can be factored out into a number of finite-state transducers. In section 6, we explain how these transducers can be used to enable tight-coupling of multimodal language processing with speech recognition. We discuss the issue of ambiguous input and mutual compensation in section 7. In sections 8 and 9, we describe the details of a multimodal grammar for the directory assistance application and the results of tight-coupling pen gesture and speech modes on speech recognition experiment in this application.

https://doi.org/10.1017/S1351324904003572 Published online by Cambridge University Press

2 Previous work

Systems capable of accepting combinations of spoken or typed input with pen or hand gestures have existed since at least the early 1980s. Carbonell (1970) described a geography tutor that accepted both typed input and mouse input. The 'put that there' system (Bolt 1980) combined spoken input with gestures made at a map display using a 3D mouse. CUBRICON (Neal and Shapiro 1991) provides spoken and mouse input to a map display. The XTRA system (Allgayer, Harbusch, Kobsa, Reddig, Reithinger and Schmauks 1989) provided a multimodal interface for help with tax forms. Koons, Sparrell and Thorisson (1993) present multimodal interfaces that support parallel input, from gestures, gaze, and speech. Cohen, Johnston, McGee, Oviatt, Pittman, Smith, Chen and Clow (1998) describe QuickSet, a mobile system which supports synchronous input by speech and pen gestures for controlling distributed interactive simulations. Cheyer and Julia (1995) present a map-based pen/voice interface for accessing information about hotels. Vo (1998) developed a toolkit for building speech/pen multimodal interfaces, that has been used to build a city guide application (QuickTour), a multimodal appointment scheduler (Jeanie-II), and a multimodal football simulator (Quarterback). The SmartKom project (Wahlster 2002) addresses multimodal interaction using combinations of speech with pen or hand gesture, in mobile, home, and public kiosk settings. Bellik (1995) describes a multimodal interface controller using augmented transition networks which extends the use of transition networks to represent user interfaces (Jacob 1986). This approach has been applied to a user interface for a multimodal window manager (Bellik and Teil 1993) and a multimodal tool for manipulation of text by blind users (Bellik and Burger 1994).

There is also a growing body of work on generation of output involving multiple modes. This typically involves generating text or speech in combination with static or dynamic graphical displays. For example, MAGIC (Dalal, Feiner, McKeown, Pan, Zhou, Hollerer, Shaw, Feng and Fromer 1996) combines speech generation with graphical displays to provide information on a patient's condition while WIP (Wahlster, André, Finkler, Profitlich and Rist 1993) and PPP (André, Muller and Rist 1996) provide depictions of three-dimensional objects combined with text or speech. We refer the reader to André (2002) for a detailed overview of work in both multimodal input processing and multimodal output generation.

2.1 Unification-based multimodal understanding

To achieve an effective method for integration of content distributed over multiple modes, Johnston (1998b; 1998a) uses techniques from natural language processing (unification-based grammars and chart parsing) to support parsing and interpretation of multimodal utterances. In that approach, speech and gesture recognition produce N-best lists of recognition results which are assigned typed feature structure representations (Carpenter 1992). These are passed to a multidimensional chart parser that uses a multimodal unification-based grammar to combine the representations assigned to the input elements. Possible multimodal interpretations are then ranked and the optimal interpretation is passed on for execution. The

https://doi.org/10.1017/S1351324904003572 Published online by Cambridge University Press

traditional chart parsing algorithm operates on a linear sequence of input elements. Multidimensional chart parsing is a generalization of traditional chart parsing which enables parsing of inputs which do not form a discrete linear sequence in a single dimension (Johnston 1998b). This approach overcomes many of the limitations of previous approaches to multimodal integration, such as Bolt (1980) and Neal and Shapiro (1991), as described in Johnston, Cohen, McGee, Oviatt, Pittman and Smith (1997, p. 282).

Unification-based multimodal parsing supports combinations of speech with multiple gestures and visual parsing of unimodal commands consisting of sequences of gestures. Furthermore, its declarative nature facilitates rapid prototyping and iterative development of multimodal systems. Also, the unification-based approach allows for mutual compensation of recognition errors in the individual modalities. For example, the selected multimodal parse might contain the third-best speech recognition result and second-best gesture recognition result. Oviatt (1999) demonstrates statistically significant mutual compensation effects in processing speech/pen input using the unification-based architecture in Johnston *et al.* (1997). This architecture precedes Johnston (1998b; 1998a) and only supports combinations of a single spoken element with a single gesture. While Johnston (1998b) provides a declarative representation of multimodal integration strategies, in Johnston *et al.* (1997) the multimodal integration strategy is encoded procedurally within a multimodal integration algorithm.

However, the unification-based approach does not allow for tight-coupling of multimodal parsing with speech and gesture recognition. Compensation effects are dependent on the correct answer appearing in the *N*-best list of interpretations assigned to each mode. Multimodal parsing cannot directly influence the progress of speech or gesture recognition. The multidimensional parsing approach is also subject to significant concerns in terms of computational complexity. In the worst case, the multidimensional parsing algorithm (Johnston 1998a, p. 626) is exponential with respect to the number of input elements. Also, this approach does not provide a natural framework for combining the probabilities of speech and gesture events in order to select among multiple competing multimodal interpretations. Wu, Oviatt and Cohen (1999) present a statistical approach for selecting among multiple possible combinations of speech and gesture. However, it is not clear how the approach will scale to more complex verbal language and combinations of speech with multiple gestures. The unification-based approach also runs into significant problems with respect to choosing among multiple competing parses and interpretations. Probabilities associated with composing speech events and multiple gestures need to be combined. Unimodal interpretations need to be compared to multimodal interpretations and so on. This can all be achieved but involves significant post processing of sets of competing multimodal interpretations generated by the multimodal parser.

Johnston (2000) shows how the assumption in the application of unification-based parsing to multimodal integration (Johnston 1998b) that speech parsing precedes multimodal parsing results in unnecessary complication of the speech grammar. As the speech grammar is extended to support conjunction, deictic numeral expressions

https://doi.org/10.1017/S1351324904003572 Published online by Cambridge University Press



Fig. 1. PDAMVPQ: Multimodal corporate directory and messaging user interface client on a Fujitsu wireless tablet computer.

(e.g. *"these four"*) and so on, it becomes increasingly difficult to build an appropriate multimodal subcategorization frame indicating the list of gestures required and the constraints on them. In that paper, Johnston proposes an architecture in which speech parsing and multimodal parsing are separate modules which collaborate in order to parse and integrate multimodal inputs. Speech parsing proceeds until a phrase such as *"this person"* has been parsed and assigned a feature structure representation. This fragment is then passed to the multimodal parser which combines the fragment with an appropriate gesture. The result is then passed back to the speech parsing component so that parsing can continue. This communication back and forth continues until the input is completely parsed and integrated with gesture. In the approach presented here the complexity of the interface between speech parsing and multimodal integration is avoided by combining the two in a single grammar.

3 Multimodal directory access application

In this paper, we illustrate our approach with examples from a prototype system that provides multimodal corporate directory access and messaging. In this system users interact with a visual representation of a company directory using combinations of speech and pen input. For example, the user can say *"email this person and this person"* and point with the pen at pictures of two people on the user interface display. The system would then bring up an email message window with the appropriate addressees and allow the user to write an email message using handwriting. The multimodal interface runs on a Fujitsu pen computer (Figure 1). This system is a

https://doi.org/10.1017/S1351324904003572 Published online by Cambridge University Press

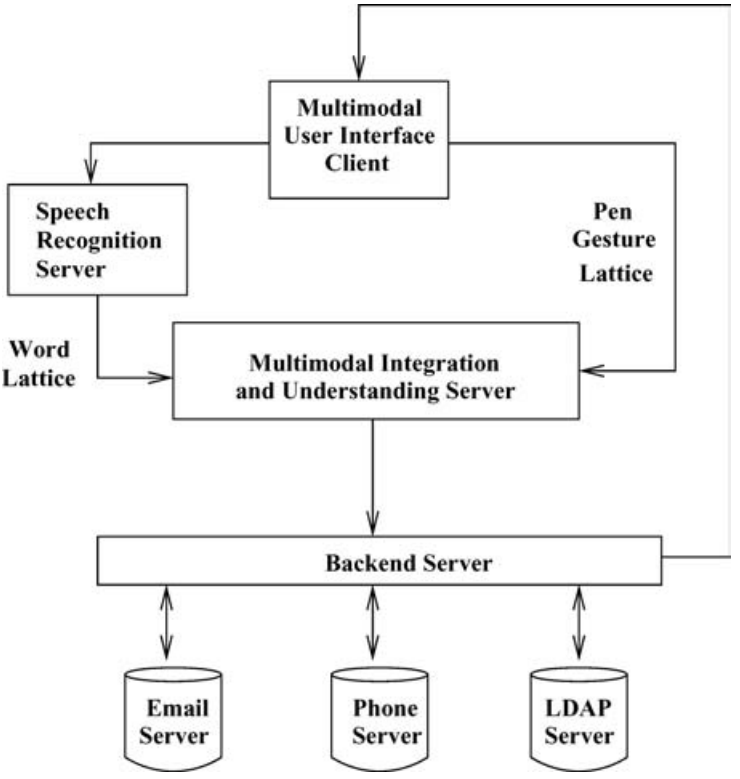


Fig. 2. Architecture for multimodal corporate directory and messaging application.

multimodal variant of the Voice POST Query system (VPQ) (Buntschuh, Kamm, DiFabbrizio, Abella, Mohri, Narayanan, Zeljkovic, Sharp, Wright, Marcus, Shaffer, Duncan and Wilpon 1998).

The system consists of a number of components which communicate with each other over TCP/IP sockets (Figure 2). In addition to the multimodal user interface client (Figure 1), the system uses a speech recognition server, a multimodal integration and understanding server, and a backend server which accesses a POST database of personnel information (LDAP server), a phone server which is able to initiate and manage telephone calls, and an email server. The recognition, multimodal integration, and backend servers can reside either on the device or be made available over a wireless network. The multimodal user interface client is responsible for receiving the user's speech and pen inputs. Speech is passed to the recognition server which yields a lattice representing the possible recognized strings. Pen gestures are collected by the multimodal user interface and composed into a lattice representation. These two lattices are received by the multimodal integration and understanding component, which fuses the two input modes and determines the most likely application command that the user is trying to convey. This is passed to the backend server for execution. Some commands result in a database lookup and presentation of information to the user on the multimodal user interface client.

https://doi.org/10.1017/S1351324904003572 Published online by Cambridge University Press

Other commands such as *"call this person"* result in the initiation of a telephone call through the phone server. Our focus in this paper is on the multimodal integration and understanding server.

4 Finite-state language processing

Finite-State Transducers (FST) are Finite-State Automata (FSA) where each transition consists of an input and an output symbol. The transition is traversed if its input symbol matches the current symbol in the input and generates the output symbol associated with the transition. In other words, an FST can be regarded as a 2-tape FSA with an input tape from which the input symbols are read and an output tape where the output symbols are written.

Finite-state machines have been extensively applied to many aspects of language processing including: speech recognition (Pereira and Riley 1997; Riccardi, Pieraccini and Bocchieri 1996), phonology (Kaplan and Kay 1994), morphology (Koskenniemi 1984), chunking (Abney 1991; Joshi and Hopely 1997; Bangalore 1997), parsing (Roche 1999), and machine translation (Bangalore and Riccardi 2000).

Finite-state models are attractive mechanisms for language processing since they are (a) efficiently learnable from data (b) generally effective for decoding and (c) associated with a calculus for composing machines which allows for straightforward integration of constraints from various levels of language processing. Furthermore, software implementing the finite-state calculus is available for research purposes (van Noord 1997; Mohri, Pereira and Riley 1998). Another motivation for our choice of finite-state models is that they enable tight integration of language processing with speech and gesture recognition.

5 Finite-state multimodal grammars

Multimodal integration involves merging semantic content from multiple input modes to build a joint interpretation for a multimodal utterance. We use a finite-state device to parse inputs from multiple modes and to combine their content into a single semantic representation. For an interface with n modes, a finite-state device operating over $n + 1$ tapes is needed. The first n tapes represent the input modes and $n + 1$ is an output stream representing their composition. In the case of speech and pen input there are three tapes, one for speech, one for pen gesture, and a third for their combined meaning.[1]

Figure 3 provides a graphical visualization of this three tape finite-state device which reads the speech and gesture inputs on the two input tapes and writes out their possible combined meaning representations on the third tape. The vertical bar represents the three tape machine. The device has two read heads and one write

[1] In a different application, Kay uses a finite-state device with four tapes in order to capture non-concatenative morphology in Arabic (Kay 1987). Kiraz and Grimley-Evans (1998) present a prolog implementation of a multi-tape finite-state device for non-concatenative morphology. There have been applications of multi-tape finite-state devices in computational phonology as well (Kornai 1995; Bird and Ellison 1994).

https://doi.org/10.1017/S1351324904003572 Published online by Cambridge University Press

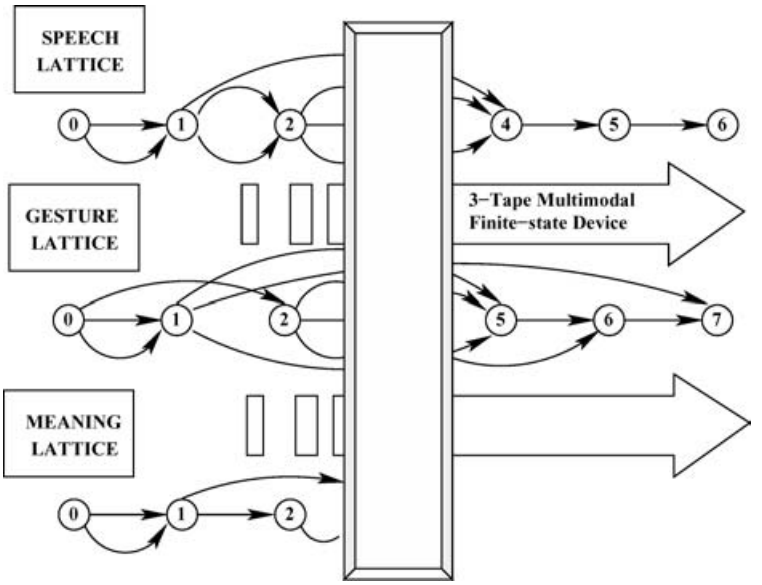


Fig. 3. Three tape multimodal finite-state device.

S	→	V NP ε:ε:])	NP	→	DET N
CONJ	→	and:ε:,	NP	→	DET N CONJ NP
V	→	email:ε:email([	DET	→	this:ε:ε
V	→	page:ε:page([	DET	→	that:ε:ε
N	→	person:G_p:person(ENTRY			
N	→	organization:G_o:org(ENTRY			
N	→	department:G_d:dept(ENTRY			
ENTRY	→	ε:e_1:e_1 ε:ε:)			
ENTRY	→	ε:e_2:e_2 ε:ε:)			
ENTRY	→	ε:e_3:e_3 ε:ε:)			
ENTRY	→	...			

Fig. 4. Multimodal grammar fragment.

head. In general, the speech and gesture inputs are lattices and their combined meaning representation is a lattice of meaning symbols.

As an example, in the messaging application described above, users issue spoken commands such as *"email this person and that organization"* and gesture at the appropriate person and organization on the screen. The structure and interpretation of multimodal commands of this kind can be captured declaratively in a multimodal context-free grammar. We present a multimodal grammar fragment capable of handling such commands in Figure 4. This is a single grammar of multimodal language which describes the relationship between speech inputs, pen gesture inputs, and their combined meaning.

The non-terminals in the multimodal grammar are atomic symbols. The multimodal aspects of the grammar become apparent in the terminals. Each terminal

https://doi.org/10.1017/S1351324904003572 Published online by Cambridge University Press

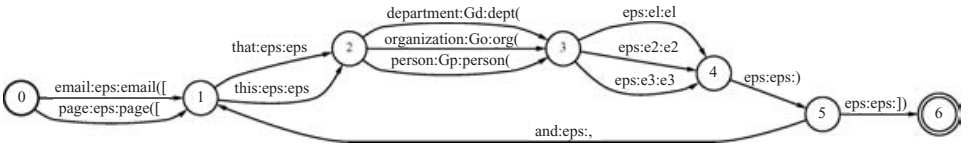


Fig. 5. Multimodal three-tape FSA.

contains three components $W:G:M$ corresponding to the $n+1$ tapes, where W is for the spoken language stream, G is the gesture stream, and M is the combined meaning. The epsilon symbol (ε or eps) is used to indicate when one of these is empty in a given terminal. The symbols in W are words from the speech stream. The symbols in G are of two types. Symbols like G_o indicate the presence of a particular kind of gesture in the gesture stream, while those like e_1 are used as references to entities referred to by the gesture (see section 5.3). Simple deictic pointing gestures are assigned semantic types based on the entities they are references to. G_p represents a gestural reference to a person on the display, G_o to an organization, and G_d to a department. Compared with a feature-based multimodal grammar, these types constitute a set of atomic categories which make the relevant distinctions for gesture events predicting speech events. For example, if the gesture is G_p then phrases like *this person* and *him* are preferred speech events. These categories also play a role in constraining the semantic representation when the speech is underspecified with respect to semantic type (e.g. *"email this one"*). These gesture symbols can be organized into a type hierarchy reflecting the ontology of the entities in the application domain. For example, there might be a general type G with subtypes G_o and G_p, where G_p has subtypes G_{pm} and G_{pf} for male and female.

A multimodal CFG (MCFG) can be defined formally as quadruple $< N, T, P, S >$. N is the set of nonterminals. P is the set of productions of the form $A \rightarrow \alpha$ where $A \in N$ and $\alpha \in (N \cup T)^*$. S is the start symbol for the grammar. T is the set of terminals of the form $(W \cup \{\varepsilon\}) : (G \cup \{\varepsilon\}) : M^*$ where W is the vocabulary of speech, G is the vocabulary of gesture=$GestureSymbols \cup EventSymbols$; $GestureSymbols = \{G_p, G_o, G_{pf}, G_{pm}, \ldots\}$, along with a finite collection of $EventSymbols = \{e_1, e_2, \ldots, e_n\}$. M is the vocabulary to represent meaning and includes event symbols ($EventSymbols \subset M$).

In general, a context-free grammar can be approximated by an FSA (Pereira and Wright 1997; Nederhof 1997). The transition symbols of the approximated FSA are the terminals of the context-free grammar and in the case of multimodal CFG as defined above, these terminals contain three components, W, G and M. The multimodal CFG fragment in Figure 4 translates into the FSA in Figure 5, a three-tape finite state device capable of composing two input streams into a single output semantic representation stream.

5.1 Contradictory inputs

The multimodal grammar we presented above only allows for non-contradictory input from the user. This means that if the user's input is contradictory or

https://doi.org/10.1017/S1351324904003572 Published online by Cambridge University Press

ill-formed, for example, they say "*tell me about these three people*" and they point at only two people, the grammar will either not generate a meaning or possibly find another path in the lattice which does match the gesture. The gesture is taken to be more reliable than speech (which makes sense since the gestures here are pointing gestures only and do not involve recognition). Ideally the grammar would enable recognition of both contradictory and non-contradictory inputs, perhaps with negative weights on the contradictory paths (adopting a kind of back-off strategy). However, there is a tension, faced in all speech and multimodal systems, between being able to recognize ill-formed inputs and the predictive power of the grammar. If you allow a spoken or multimodal grammar to recognize ill-formed inputs then its predictive capability is reduced and it may result in increased misrecognitions of well formed inputs.

5.2 Temporal constraints

Within the finite-state mechanism, our multimodal grammars do not impose explicit temporal constraints. In the unification-based approaches (Johnston 1998b; Johnston *et al.* 1997), functional constraints are used to impose temporal constraints on the combination of inputs. These constraints are applied between complete spoken utterances and pen gestures and between pen gestures and other pen gestures. In our approach, the temporal constraint between the speech as a whole and pen gestures can be applied outside of the finite-state integration and understanding framework. This is achieved using a timeout mechanism when building speech and pen gesture lattices for combination in accordance with the multimodal grammar. If a speech event and a gesture event overlap temporally or occur within a specified temporal interval of each other they will be paired. For the multimodal messaging application we found a temporal interval of between one and two seconds to be sufficient. The multimodal integration component receives notifications of voice and pen activity and extends the temporal interval and waits for additional input if there is ongoing activity in one of the modes. For example, if the user has started drawing by touching the pen to the screen, the multimodal integration component receives a notification and it will not proceed with multimodal parsing until the resulting pen gesture lattice is received. Beyond this high level temporal constraint on the combination of speech and gesture, in multi-gesture utterances the primary function of temporal constraints is to force an order on pen gestures. If a user says *"move this here"* and makes two pointing gestures, the first corresponds to *"this"* and the second to *"here"*. This order constraint is easily expressed in the finite-state framework.

Bellik (1995; 1997) provides examples indicating the importance of precise temporal constraints for proper interpretation of multimodal utterances. We abstract from the specific example in Bellik (1997), and show a generalized version in Figure 6. Here, the three events X, Y and Z appear in the same temporal order in both the scenarios, as shown in Figure 6, but result in two distinct interpretations. This example is used by Bellik to illustrate that order is insufficient to determine the interpretation of multimodal events.

https://doi.org/10.1017/S1351324904003572 Published online by Cambridge University Press

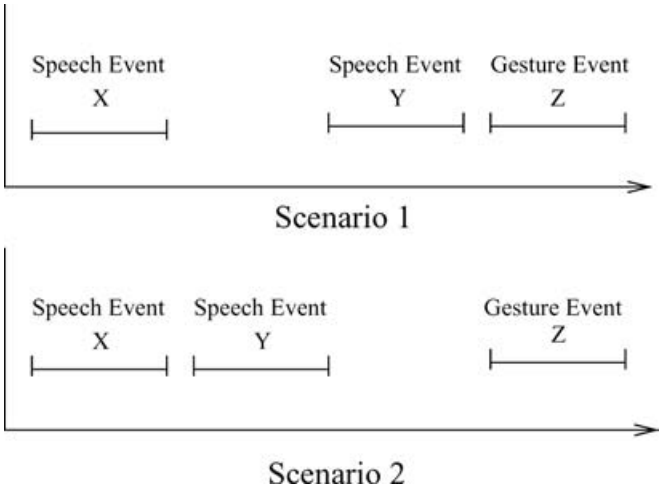


Fig. 6. Two Scenarios with the same order of events but receiving different interpretations.

In the architecture we present, we arrive at two distinct interpretations because the temporal distance between adjacent events in the two scenarios differs. In scenario 1, the speech recognizer's endpointing mechanism applies after the speech event X, ceasing recognition, and resulting in a unimodal interpretation of X. Subsequently, the events Y and Z are treated as a multimodal event which receives a multimodal interpretation. In contrast, in scenario 2, the speech recognizer's endpointing mechanism will apply after events X and Y resulting in a unimodal interpretation of X and Y together and subsequently the event Z receives a unimodal interpretation. In an application for chemical operations mentioned in Bellik (1997), event X denotes "*pressure*" and event Y denotes "*plus two*" and Z denotes "*pointing at the temperature icon*". In scenario 1, this produces the interpretation wherein the system produces a readout of the current pressure and then raises the temperature by two units. In scenario 2, the same sequence of events results in the pressure being increased by two units and then a speech readout of the current temperature. A second example in Bellik (1997) is presented with a similar notion of temporal distance resulting in two different interpretations for the same sequence of events. So, the critical fact is that Bellik's examples involve not single multimodal utterances but sequences of two utterances. We utilize the finite-state multimodal grammar to integrate and understand single multimodal utterances, not sequences of multimodal utterances.

The result of gesture and speech recognizer timeouts is to produce a notion of a dialog turn. The multimodal framework attempts an interpretation for all events that occur within each turn. The combination of interpretations across turns is the task of a multimodal dialog manager and is not addressed in this paper.

5.3 Finite-state meaning representation

Implementations of syntactic and semantic models of language typically utilize more than finite-state computational power. In this research we are interested in modelling

https://doi.org/10.1017/S1351324904003572 Published online by Cambridge University Press

the subset of a language needed for a broad class of interactive applications, which can be modelled with a finite state device. While work in finite-state language processing has addressed computational syntax (Krauwer and Tombe 1981; Gross 1989, 1997), it has not addressed the assignment of meaning to strings. The approach we represent here is novel in two respects. First, in its utilization of a finite-state mechanism to translate strings of words into semantic representations for the subset of the language we are interested in and second, in its application of finite-state methods to combining multimodal inputs.

Our basic approach is to write symbols onto the third tape, which when concatenated together yield the semantic representation for the multimodal utterance. It suits our purposes here to use a simple logical representation with predicates *pred(....)* and lists *[a,b,...]*. Many other kinds of semantic representation could be generated. In the fragment in Figure 4, the word *"email"* contributes *email*([to the semantics tape, and the list and predicate are closed when the rule $S \rightarrow V\ NP\ \varepsilon{:}\varepsilon{:}]$) applies. The word *"person"* writes *person* on the semantics tape.

We use a finite-state device to produce a semantic representation language that contains parentheses. It is well known that a finite-state device is insufficient to ensure the well-formedness of nesting parentheses to arbitrary depths. However, in our application, we solve this problem by limiting the depth of parentheses embedding in the resulting semantic expressions to a bounded depth. It is interesting that even with this restriction, we are able to encode the semantic representations needed for a large class of interactive applications.

A significant problem we face in adding meaning into the finite-state framework is how to represent all of the different possible specific values that can be contributed by a gesture. For deictic references a unique identifier is needed for each object in the interface that the user can gesture on. For example, if the interface shows lists of people, there needs to be a unique identifier for each person. As part of the composition process this identifier needs to be copied from the gesture stream into the semantic representation. In the unification-based approach to multimodal integration, this is achieved by feature sharing (Johnston 1998b, 1998a). In the finite-state approach, we would need to incorporate all of the different possible IDs into the FSA. For a person with id *objid*345 you need an arc $\varepsilon{:}objid345{:}objid345$ to transfer that piece of information from the gesture tape to the meaning tape. All of the arcs for different IDs would have to be repeated everywhere in the network where this transfer of information is needed. Furthermore, these arcs would have to be updated as the underlying database was changed or updated. Matters are even worse for more complex pen-based data such as drawing lines and areas in an interactive map application (Cohen *et al.* 1998). In this case, the coordinate set from the gesture needs to be incorporated into the semantic representation. It might not be practical to incorporate the vast number of different possible coordinate sequences into an FSA.

Our solution to this problem is to store these specific values associated with incoming gestures in a finite set of variables labeled $e_1,e_2,e_3,\ldots$ and in place of the specific content write in the name of the appropriate variable in the gesture lattice. Instead of having the specific values in the FSA, we have the transitions

https://doi.org/10.1017/S1351324904003572 Published online by Cambridge University Press

W: email	this person	and	that organization
G:	G_p e_1		G_o e_2
M: email([	person(e_1)	,	org(e_2)])

Fig. 7. Messaging domain example (PDAMVPQ).

```
NP      →  ε:ε:manager(  POSSP2  manager:ε:)
NP      →  ε:ε:secretary(  POSSP2  secretary:ε:)
POSSP2  →  ε:ε:lab(  POSSP  lab's:ε:)
POSSP2  →  ε:ε:dept(  POSSP  department's:ε:)
POSSP   →  his:Gp:ε  ENTRY
POSSP   →  this::ε:ε  person's:eps:person(  ε:Gp:ε  ENTRY  ε:ε:]
```

Fig. 8. Grammar fragment for possessives.

$\varepsilon{:}e_1{:}e_1$, $\varepsilon{:}e_2{:}e_2$, $\varepsilon{:}e_3{:}e_3 \ldots$ in each location where content needs to be transferred from the input gesture lattice to the output meaning lattice (see Figure 5). These are generated from the *ENTRY* productions in the multimodal CFG in Figure 4. The gesture interpretation module empties the buffers and starts back at e_1 after each multimodal command, and so we are limited to a finite set of gesture events in a single utterance. Returning to the example *"email this person and that organization"*, assume the user gestures on entities *objid*367 and *objid*893. These will be stored in buffers e_1 and e_2. Figure 7 shows the speech and gesture streams and the resulting combined meaning.

The elements on the meaning tape are concatenated and the buffer references are replaced to yield $email([person(objid367), org(objid893)])$.

The immediate question that arises when building semantic representations using a finite-state framework is how to handle recursive structures such as possessives and conjunction. In the grammar in Figure 4 conjunction is handled by unfolding the *NP* category on the left hand side of the conjunction (Pereira and Shieber 1987). The right recursion is handled by finite-state approximation in the compilation from the grammar to the finite-state device (Figure 5) (Pereira and Wright 1997). Possessives are handled up to a certain size by introducing intermediate categories *POSSP* and *POSSP2* (Figure 8). This increases the size of the grammar and resulting machine but is manageable for a broad class of applications. For a broad class of interactive applications there is no need for more than two levels of recursion in possessive expressions. In this grammar each level of the possessive introduces another predicate. The phrase *"this person's lab's manager"* is assigned the representation *manager(lab(person(e1)))*.

Given the head-final nature of possessives in English we need to introduce the predicate corresponding to the head first then add each of the possessives in turn. This complicates the grammar since we need a rule for each noun at each level so that we can capture the dependency between the predicate and the head noun.

https://doi.org/10.1017/S1351324904003572 Published online by Cambridge University Press

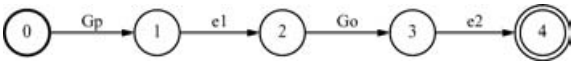


Fig. 9. *Gesture* finite-state machine.

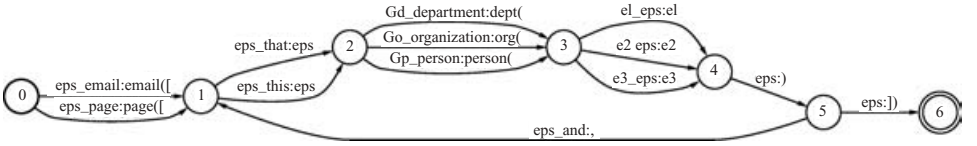


Fig. 10. Transducer relating gesture and speech to meaning ($\mathscr{T}$:$(G \times W) \rightarrow M$).

Staying with approximately the same semantic representation, one way to simplify the grammar is to allow for predicates to appear on either side of the brackets. For example, *"this person's lab's manager"* could be *((person(e1))lab)manager*. This more closely matches the syntactic form and enables a simpler grammar where the semantic predicate is introduced with the head word; without the need for a rule for each noun at each level.

As more recursive phenomena such as complex noun phrases are added to the grammar the resulting machines increase in size. However, the computational consequences of this can be lessened by lazy evaluation techniques (Mohri 1997) and we believe that this finite-state approach to constructing semantic representations is viable for a broad range of sophisticated language interface tasks. We have implemented a sizeable multimodal CFG for VPQ: 417 rules and a lexicon of 2388 words.

5.4 Multimodal finite-state transducers

While a three-tape finite-state automaton is feasible in principle (Rosenberg 1964), currently available tools for finite-state language processing (van Noord 1997; Mohri, Pereira and Riley 1998) only support Finite-State Transducers (FSTs) (two tapes). Furthermore, speech recognizers typically do not support the use of a three-tape FSA as a language model. To implement our approach, we convert the three-tape FSA (Figure 5) into an FST, by decomposing the transition symbols into an input component ($G \times W$) and output component M, thus resulting in a function, $\mathscr{T}$:$(G \times W) \rightarrow M$. This corresponds to a transducer in which gesture symbols and words are on the input tape and the meaning is on the output tape (Figure 10). The domain of this function $\mathscr{T}$ can be further curried resulting in a transducer $\mathscr{R}$:$G \rightarrow W$ that maps from gestures G to work W (Figure 11). This transducer captures the constraints that gesture places on the speech stream and we use it as a language model for constraining the speech recognizer based on the recognized gesture string. In the following section, we explain how $\mathscr{T}$ and $\mathscr{R}$ are used in conjunction with the speech recognition engine and gesture recognizer and interpreter to parse and interpret multimodal input.

https://doi.org/10.1017/S1351324904003572 Published online by Cambridge University Press

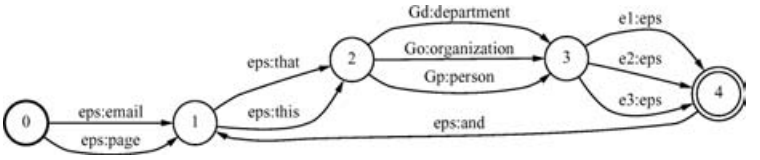


Fig. 11. Transducer relating gesture and speech ($\mathscr{R}$:$G \rightarrow W$).

6 Applying multimodal transducers

There are number of different ways in which multimodal finite-state transducers can be integrated with speech and gesture recognition. The best approach to take depends on the properties of the particular interface to be supported. One approach we outline here involves recognizing gesture first then using the observed gestures to modify the language model for speech recognition. This is a good choice if there is limited ambiguity in gesture recognition, for example, if the majority of gestures are unambiguous deictic pointing gestures as in the directory and messaging application discussed here. There can however be consequences for system latency since this approach requires waiting until after the gesture inputs have been processed to start speech recognition. Another alternative is to always use the same speech recognition grammar, one which accepts all of the possible speech inputs enumerated by the multimodal grammar, take lattice output from the speech recognizer, and through multimodal integration use gesture information to rule out incompatible speech strings from the lattice. We describe both alternatives below.

Our approach is implemented using the AT&T finite-state transducer software library.[2] This library provides basic operations for manipulation of finite-state machines and finite-state transducers. Our approach essentially involves a cascade of finite-state operations with a small amount of additional code to factor a three-tape machine into a two-tape machine and perform other minor intermediate steps. In what follows we first present the approach at a high level then provide pseudocode showing the sequence of finite-state operations required.

The first step is for the gesture recognition and interpretation module to process incoming pen gestures and construct a finite state machine *Gesture* corresponding to the range of gesture interpretations. If the gestural input involves gesture recognition or is otherwise ambiguous, as discussed above, it is represented as a lattice indicating all of the possible recognitions and interpretations of the gesture stream. This allows speech to compensate for gesture errors and allows for mutual compensation. In our example case (Figure 7) the gesture input is unambiguous and the *Gesture* finite state machine will be as in Figure 9.

This *Gesture* finite state machine is then composed with the transducer $\mathscr{R}$ which represents the relationship between speech and gesture (Figure 11). The result of this composition is a transducer *GestLang* (Figure 12). This transducer represents

[2] See `http://www.research.att.com/sw/tools/fsm/` for details. Other toolkits for finite-state language processing include van Noord (1997).

https://doi.org/10.1017/S1351324904003572 Published online by Cambridge University Press

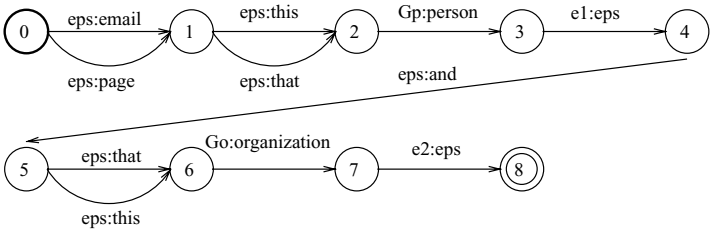


Fig. 12. GestLang transducer.

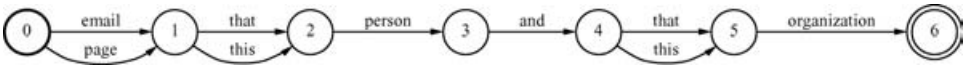


Fig. 13. Projection of Output tape of GestLang transducer.

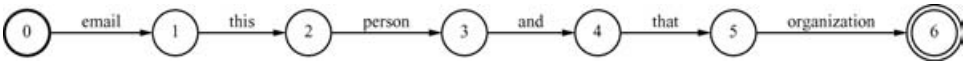


Fig. 14. Result from speech recognizer.

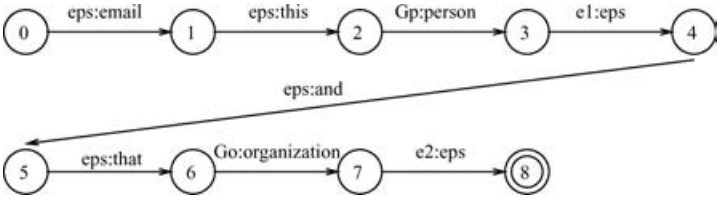


Fig. 15. GestureSpeech FST.

the relationship between this particular stream of gestures and all of the possible word sequences that could co-occur with those gestures. It also aligns the speech and gesture input so it can be later used with the meaning transducer. In order to use this information to guide the speech recognizer, we then take a projection on the output tape (speech) of *GestLang* to yield a finite-state machine which is used as a language model for speech recognition (Figure 13). Using this model enables the gestural information to directly influence the speech recognizer's search. Speech recognition yields a lattice of possible word sequences. In our example case it yields the word sequence *"email this person and that organization"* (Figure 14). We now need to reintegrate the gesture information that we removed in the projection step before recognition. This is achieved by composing *GestLang* (Figure 12) with the result lattice from speech recognition (Figure 14), yielding transducer *GestSpeechFST* (Figure 15). This transducer contains the information both from the speech stream and from the gesture stream. The next step is to generate the combined meaning representation. To achieve this *GestSpeechFST* ($G : W$) is converted into an FSM *GestSpeechFSM* by combining output and input on one tape ($G \times W$) (Figure 16).

https://doi.org/10.1017/S1351324904003572 Published online by Cambridge University Press

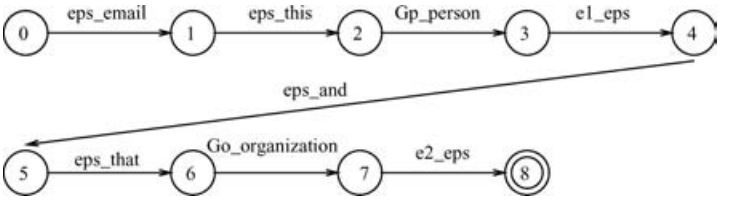


Fig. 16. GestureSpeech FSM.

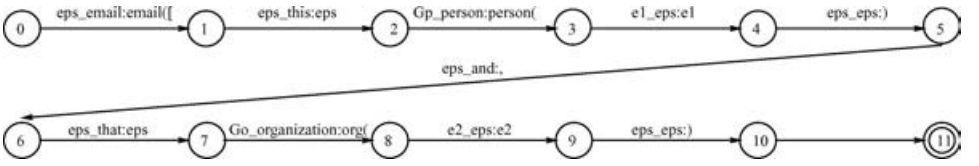


Fig. 17. Result transducer.

1. Build lattice representing possible interpretations of gesture G' (Figure 9).
2. Abstract over specific content in G', store in set of buffers $e_1 \ldots e_n$
3. Compose gesture lattice G' (Figure 9) with $G : W$ (Figure 11) transducer yielding $G' : W'$ (Figure 12)
4. Project on W' side (Figure 13) of $G' : W'$ transducer
5. Perform speech recognition using W' as the language model yielding string W'' (Figure 14)
6. Compose W'' (Figure 14) with $G' : W'$ (Figure 12) yielding $G' : W''$ (Figure 15)
7. Factor transducer $G' : W''$ (Figure 15) into an FSM G_W (Figure 16)
8. Compose G_W (Figure 16) with $G_W : M$ (Figure 10)
9. Project on output yielding meaning lattice M
10. Read off highest scoring path from M
11. Concatenate symbols on path to yield meaning representation
12. Replace buffer references $e_1 \ldots e_n$ with specific content

Fig. 18. Finite-state multimodal processing cascade – Early integration.

GestSpeechFSM is then composed with $\mathcal{T}$ (Figure 10), which relates speech and gesture to meaning, yielding the result transducer *Result* (Figure 17). The meaning is read from the output tape yielding $email([person(e_1), org(e_2)])$. We have implemented this finite-state approach and applied it in a multimodal interface to VPQ (Buntschuh *et al.* 1998) on a Fujitsu wireless PDA (Figure 1). Figure 18 lists the sequence of steps involved in the process of finite-state integration and understanding for early integration. In the early integration case information from the pen gesture input mode (G') is used to dynamically rebuild the language model used for speech recognition (W'). Figure 19 shows the late integration procedure that can be used if lattice output is available from the speech recognizer. In the late integration approach a more general purpose language model (W) is used for speech recognition and the information from the pen gesture input mode (G') is effectively used to rescore the lattice resulting from speech recognition. Steps 7–12 are the same in both the early and late integration approaches. The late integration approach has the significant

https://doi.org/10.1017/S1351324904003572 Published online by Cambridge University Press

1. Perform speech recognition with W (Figure 13) projection of $G : W$ (Figure 12), result is W' (Figure 14)
2. Build lattice representing possible interpretations of gesture G' (Figure 9)
3. Abstract over specific content in G', store in set of buffers $e_1 \ldots e_n$
4. Compose gesture lattice G' (Figure 9) with $G : W$ (Figure 11) machine yielding $G' : W$ (Figure 12)
5. Compose $G' : W$ (Figure 12) with W' (Figure 14) yielding $G' : W'$ (Figure 15)
6. Factor transducer $G' : W''$ (Figure 15) into FSM G_W (Figure 16)
7. Compose G_W (Figure 16) with $G_W : M$ (Figure 10)
8. Project on output yielding meaning lattice M
9. Read off highest scoring path from M
10. Concatenate symbols on path to yield meaning representation
11. Replace buffer references $e_1 \ldots e_2$ etc with specific content

Fig. 19. Finite-state multimodal processing cascade – Late integration.

advantage over the early integration approach that speech recognition can commence as soon as the user starts to speak. In the early integration approach, speech recognition cannot proceed until the pen input is complete, resulting in increased system latency.

7 Ambiguity and compensation

In the application described here the pen input mode involves only unambiguous pointing gestures at people or organizations on a graphical display. In other applications pen inputs can be highly ambiguous. Drawn words or symbols may be subject to recognition errors. In applications with spatial displays such as maps (Johnston, Bangalore, Vasireddy, Stent, Ehlen, Walker, Whittaker and Maloor 2002) drawing an area on the map can be ambiguous between referring to a spatial location and selecting an entity which lies within the area. Even simple deictic pointing gestures can be highly ambiguous. For example, in a text editing application if the user points at a character it is not clear whether they are referring to that character, to the word it is part of, or the paragraph. Similarly in a map-based application if the user points at a building in Central Park, they could be referring to the building, to the park, to Manhattan and so on. This problem has a long history in the philosophical literature. (See, for example, Quine's discussion of the inscrutability of reference (Quine 1969).)

Our whole framework is set up to account for ambiguity in both the speech stream and in the gesture stream. Like speech, gesture can be represented as a lattice of possibilities. For example, in a text editing application when the user points at a character/word/paragraph we would allow for all of these different interpretations in the gesture lattice. Then through combination with the speech some of this ambiguity could be resolved. For example, if the user says 'hanging indent' they are clearly trying to refer to the paragraph, if they say 'delete this word' they are referring to the word. If multimodal integration does not resolve the ambiguity a number of other options are available. For example, the system could recognize the ambiguity and enter into a confirmation subdialog with the user to resolve the

https://doi.org/10.1017/S1351324904003572 Published online by Cambridge University Press

ambiguity. This brings us into the area of multimodal dialog management, which we do not address in this paper. The application we describe here is more tool-like rather than conversational and does not use a separate dialog manager component. In some cases, one combination may have a far higher prior probability than the other and it may be safe to choose on this basis.

In addition to the application described here we have also applied the finite-state approach to a more complex application (Johnston *et al.* 2002) involving ambiguous pen-based input. In that application, the ambiguity of pen gestures is resolved through multimodal combination.

The finite-state multimodal processing architecture we have described supports mutual compensation among the modes; that is, speech can potentially correct for gesture errors and gesture can correct for speech errors. It is even possible for speech and gesture to both undergo compensation in single multimodal utterance (Oviatt 1999). However, in the application we describe here, the gesture input is made using a pen to push buttons on a tablet display and there is no ambiguity in the pen gesture inputs. Since the deictic pen gestures the user can make are unambiguous there is only potential for unidirectional compensation for speech recognition errors on the basis of pen gesture inputs. As an example, if the user points at two people on the display, then it is more likely that they are saying something like *"can you please email this person and this person"* or *"what are the phone numbers for these two people"* rather than *tell me about these four organizations"* or *"call this person"*. In the experiments described in the following section, we examine the extent to which knowledge of what the user has pointed to can improve ASR word and sentence accuracy.

8 The effect of finite-state multimodal language processing on speech recognition performance

One of our central goals in applying finite-state techniques to multimodal language processing is to enable tight coupling with a speech recognizer so that gestural information can directly constrain speech recognition search. To evaluate our approach, we designed an experiment which examines the extent to which the early integration of gestural information enabled by our approach can improve speech recognition performance. This involved development of a realistic multimodal grammar, collection of a test corpus of multimodal speech data, a series of different speech recognition experiments on that corpus, and analysis of the results.

8.1 Multimodal grammar development

Our intention was to create a grammar of sufficient size and complexity that it could support a realistic and useful task. We built a grammar for the multimodal messaging application which contains 417 rules and 2388 different words. It is called PDAMVPQ and constitutes a multimodal extension of the grammar used for VPQ, a corporate directory information service (Buntschuh *et al.* 1998). A broad range of different messaging commands are supported including calling, faxing, and email, in addition to commands to find directory information about individuals

https://doi.org/10.1017/S1351324904003572 Published online by Cambridge University Press

1. email these projects and this department's secretary
2. send a fax to this person's department and these two people
3. I'd like to talk to this person
4. page Kamm's group's secretary and that person
5. could you please send a message to these four departments
6. his mobile phone number please
7. I'm looking for the email address for this lab please
8. I'd like to talk with this person's manager please
9. send email to these three departments and this center
10. can you please email Srinivas Bangalore and that department

Fig. 20. Sample multimodal directory and messaging commands.

and organizations. The grammar supports deictic expressions, proper names (400), conjunction, possessives, and numeral expressions. Figure 20 provides examples of a number of different commands supported by the grammar.

8.2 Data collection

We collected a set of one hundred sample commands from each of ten volunteer subjects. The subject pool contained six men and four women, five native and five non-native speakers of English. The sample commands were drawn from the multimodal messaging domain described above. There were 891 words in the test corpus and an average sentence length of approximately nine words. We set up a data collection environment in which each subject was prompted with the sequence of one hundred sample commands. They issued each command in turn and their speech signal was recorded. We recorded speech at 8000Hz using a Sennheiser noise-cancelling microphone. In our multimodal messaging application, gesture input is limited to deictic pointing events, so there is little room for gesture recognition error. This data collection provided us with a corpus of 1000 spoken utterances and their associated gestures across 10 speakers.[3]

9 Speech recognition experiments

We used an HMM-based speech recognizer with an off-the-shelf acoustic model. Each speech file in the test corpus was recognized under a without-gesture condition and with-gesture condition at a range of beam widths from one to sixteen. The beam width is a measure of the set of candidates considered during search in ASR decoding. The speech recognizer accepts a finite-state machine as its language model. For the without-gesture condition we used the output tape projection of the ($\mathscr{R}$) transducer, which relates gesture and speech, as the language model. The resulting

[3] Speech collected in this way will be different in character to speech input to a live multimodal system. It is likely to be more consistent in timing and less disfluent; more like read speech and therefore likely to be easier to recognize. However, we believe that despite its limitations, this data collection still provides us with a baseline for examination of compensation effects.

https://doi.org/10.1017/S1351324904003572 Published online by Cambridge University Press

finite state acceptor contains all speech strings in the application grammar. For the with-gesture condition, we first encoded the gesture sequence for the current sample command into an FSA and composed it with the ($\mathscr{R}$) transducer. We then used the output tape projection of the resulting transducer as the language model. This restricted the language model for the with-gesture condition to only those strings compatible with the gesture stream for the sample command to be recognized.

We repeated these recognition experiments under a number of simulated noise conditions. We added two kinds of noise to the clean recorded signal: babble noise and car noise, each at three different signal-to-noise levels (30db, 20db, 10db). With a total of seven different noise conditions (clean, babble 30db, babble 20db, babble 10db, car 30db, car 20db, car 10db), the with-gesture and without-gesture conditions, 16 beam widths, and 1000 utterances, a total of 224,000 speech recognitions were performed.

For each recognition, we compared the result string to the reference string for that command, and used the standard *score* mechanism from NIST to compute word and sentence accuracy (`http://www.nist.gov/speech/tools/score_362tarZ.htm`). Sentence accuracy is more pertinent to the interface task that we describe, since it provides an indication of how many commands would have succeeded, but we also present word accuracy in keeping with practice in speech recognition research. Averages for word and sentence accuracy were calculated for each speaker and across the whole corpus in each noise condition and beam width for both with-gesture and without-gesture conditions. Average absolute error reduction, relative error reduction, and accuracy improvement for both word and sentence accuracy were calculated between the with-gesture and without-gesture conditions for individual speakers, noise conditions, and across the whole corpus.[4]

9.1 Results and analysis

9.1.1 Multimodal compensation for speech recognition errors

In order to determine the effects of tight-coupling finite-state multimodal language processing with speech recognition, we compared word and sentence accuracy for the with-gesture condition to the without-gesture condition. The graph in Figure 21 shows word and sentence accuracy for the with-gesture and without-gesture conditions for clean speech at beam widths from 3 to 16 averaged over all 1000 utterances in the corpus.

Word and sentence accuracy were found to be higher for the with-gesture condition at all beam widths. Both word and sentence accuracy for both conditions start to even off around beam width 11 or 12. A series of within-subject ANOVA tests showed these differences to be statistically significant at beam widths of three and above (all $p < 0.05$). Close to real-time performance can be achieved with this recognizer at beam widths of up to 10 or 11 and the rest of the results we will

[4] If x is with-gesture accuracy, y is without-gesture accuracy, then relative error reduction is (x-y)/100-y (improvement over total error) and accuracy improvement is (x-y)/x (improvement over accuracy).

https://doi.org/10.1017/S1351324904003572 Published online by Cambridge University Press

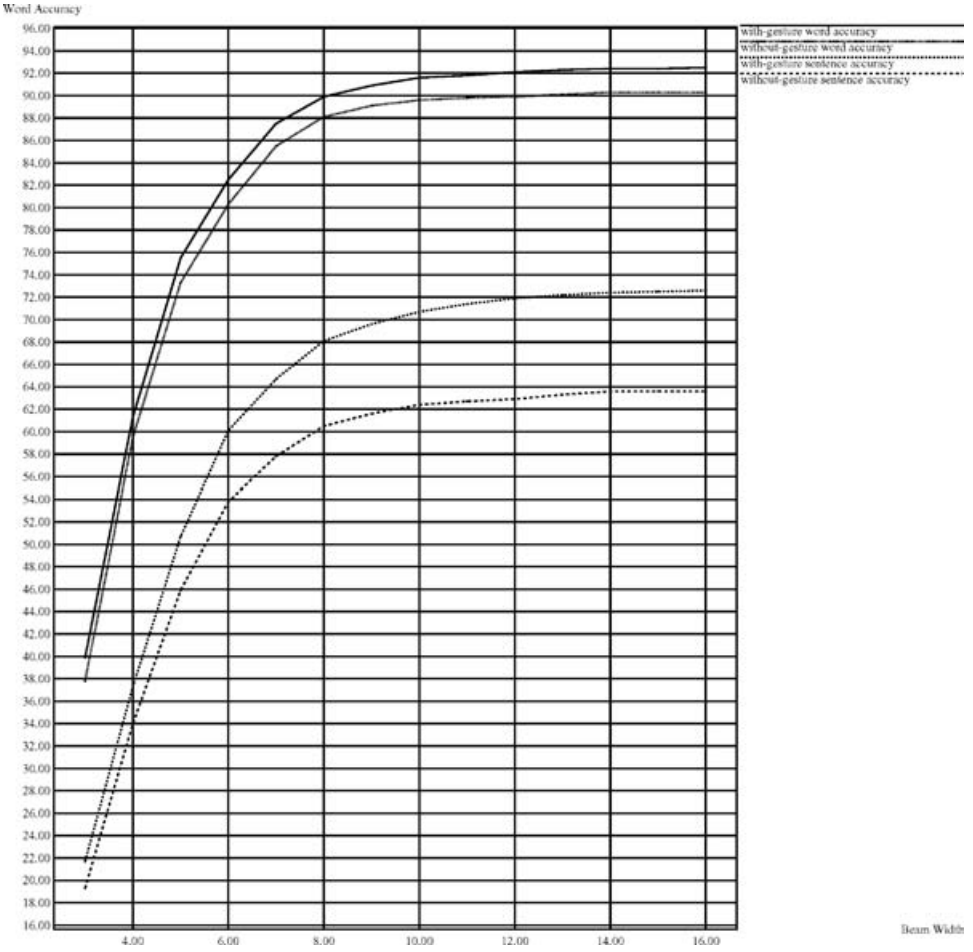


Fig. 21. Word and sentence accuracy across various beam widths for clean speech averaged across all speakers.

present below are for beam width 11. At beam width 11, average sentence accuracy is 62.7% in the without-gesture condition and 71.4% in the with-gesture condition; an absolute error reduction of 8.7% (relative error reduction of 23%). A within-subject ANOVA showed this increase in performance to be statistically significant ($F(1,9) = 63$, $p < 0.001$). At beam width 11, average word accuracy for clean speech is 89.8% without-gesture and 91.8% with-gesture; an absolute error reduction of 2% (relative error reduction of 20%). A within-subject ANOVA showed this increase in performance to be statistically significant ($F(1,9) = 49$, $p < 0.001$). Through the tight-coupling of multimodal language processing and speech recognition enabled by our finite-state approach almost a quarter (23%) of all sentence level errors were avoided.

The only other reported results we found for the role of gesture in improving speech recognition performance are given in Oviatt (1999), which reports significant levels of mutual disambiguation among the input modes in pen/voice interaction

https://doi.org/10.1017/S1351324904003572 Published online by Cambridge University Press

Table 1. *Absolute accuracy improvement with and without gesture under various noise conditions averaged across all speakers at beam width 11*

	Word accuracy		Sentence accuracy	
	w/o G	w/G	w/o G	w/G
clean	89.8	91.8	62.7	71.4
babble 30db	88.3	90.8	59.3	68.6
babble 20db	74.7	78.1	33.7	41.6
babble 10db	26.8	30	2.1	3.5
car 30db	84.8	87.4	51.5	60.9
car 20db	52.1	55	10.7	14.2
car 10db	5.7	7.4	0.1	0.1

with a dynamic map. Oviatt (1999) examines compensation in both directions, while here, given the nature of our application, we address unidirectional compensation for ASR errors. The studies are hard to compare as the tasks are very different in nature. The task in Oviatt (1999) supports a vocabulary of 400 words and a total of only 200 unique multimodal utterances each of 1–7 syllables. Our grammar has a vocabulary of 2388 words and the average test sentence length is 9 words. The architecture (and task) evaluated in Oviatt (1999) only supports combinations of speech with a single gesture while ours supports combination of speech with multiple gestures.

9.1.2 Background noise and compensation

In order to examine the relationship between background noise level and multimodal compensation, we compared word and sentence accuracy for the with-gesture and without-gesture conditions at different levels of background noise. The average word and sentence accuracy over all 1000 utterances for the seven different noise conditions at beam width 11 are given in Table 1.

This table shows how word and sentence accuracy decrease as the background noise level increases, for both babble noise and car noise. The addition of car noise had a far more pronounced effect on performance than babble noise. Speech recognition accuracy dropped a small amount with addition of babble 30db or car 30db. The car 20db drops off much more than babble 20db. At 10db, performance with babble is in the high 20s while for car it is down in the single digits. Tight-coupling of multimodal language processing and speech recognition reduced both word and sentence error at all noise levels except for sentence accuracy at car 10db which was close to zero both with and without gesture. Absolute word error reduction increases from clean to 30db to 20db for both car and babble, then decreases at 10db. Sentence error reduction increases from clean to 30db, then falls. To examine the relationship between background noise level and multimodal compensation we calculated the average accuracy improvement at each noise level as shown in Table 2. We show here only the table for babble noise (the table for car noise is similar). As the amount of noise added to the signal increases, the relative

https://doi.org/10.1017/S1351324904003572 Published online by Cambridge University Press

Table 2. *Relative accuracy improvement in different noise conditions*

	Clean	babble 30db	babble 20db	babble 10db
Relative word accuracy improvement	2.3	2.9	4.8	16
Relative sentence accuracy improvement	14.6	16.8	27.2	47.2

accuracy improvement goes up for both word and sentence accuracy. Our tightly-coupled approach continues to be effective with different levels of background noise and moreover is increasingly more effective in improving accuracy as the level of noise increases.

9.2 Efficiency considerations

To examine the effect of our approach on the efficiency and speed of speech recognition, we calculated the average time taken to recognize an utterance over all the utterances in the corpus at the 16 different beam widths for both the with-gesture and without-gesture condition. The average time climbs much more sharply in the without-gesture condition. At beam width 11, the average was 18 secs without-gesture and 10 secs with-gesture; a 44% speed-up. The size of the language model for recognition is reduced through composition with gestural information leading to a significant reduction in processing time for the with-gesture condition. This speed up is important since in order to pre-compose the gestural information, the start of speech recognition must be delayed until the gesture is made. Another helping factor is the tendency for gesture to precede speech (Oviatt 1997).

10 Discussion

The work in Oviatt (1999) utilizes the unification-based approach to multimodal integration (Johnston *et al.* 1997). In this approach, the integration component receives N-best lists of feature structures representing possible interpretations of speech and pen gesture inputs. It attempts to integrate them by determining which pairings of speech and gesture interpretations will unify. A problematic aspect of this approach to multimodal integration is that if the appropriate speech or gesture interpretation does not make it into the N-best list then there is no way for the system to determine the appropriate interpretation – the amount of compensation that is possible is inherently limited by the length of the N-best list. This problem could be overcome if the unification-based multimodal integration algorithm was redesigned and extended so that it could take take speech and gesture lattices as inputs.

The tightly coupled finite-state approach to multimodal integration has the potential to enable higher levels of compensation for speech recognition errors since it directly influences speech recognition search and does not rely on the correct answer being in the N-best list. We are examining this in ongoing work by simulating the N-best approach on this same task. In addition to the one-pass procedure we described here where gesture directly influences the speech recognition language model, the finite-state approach can also be used in a two-pass configuration where

https://doi.org/10.1017/S1351324904003572 Published online by Cambridge University Press

the multimodal finite-state transducer is used to re-score lattice output from the speech recognizer. In ongoing work, we are comparing the one and two pass approaches. However, the two-pass approach would not have the efficiency benefits described above.

Bellik (1995) presents an approach to the encoding of multimodal interaction patterns which builds on work by Jacob (1986) and others on the use of Augmented Transition Networks (ATNs) to capture the behavior of traditional graphical user interfaces. In Bellik's approach, multimodal input events such as spoken words and pointing gestures are combined in accordance with a set of compositional operators (sequential and parallel composition). Multimodal and unimodal input events are then combined using an ATN to determine the user's intention. This approach has the important property that it enables the system to give immediate incremental feedback to the user as they input the words and gestures of a single command. This assumes of course that the ASR provides the recognition results incrementally. Bellik's work is an important precursor to our approach in that it uses a transition network to represent the range of possible multimodal utterances. There are however many differences between the two approaches.

In Bellik's approach the actual combination of speech and gesture happens outside of the transition network. Sequential and parallel composition operators are used to combine click events and words. The arcs of the transition network represent words, clicks, or multimodal combinations. In our approach, the transition network itself encodes the relationship between speech and gesture and multimodal integration happens as the finite-state machines which represent speech and gesture are combined.

Furthermore, ATNs are considerably more computationally powerful than the finite-state transducers used in our approach. They allow for constraints and actions to be associated with the nodes. The computational power of the devices we use is limited to that of regular languages. As such, our approach is better suited to processing of highly uncertain input in the form of large lattices of recognition hypotheses from speech and gesture recognition. It is not clear that the ATN approach would scale as effectively.

In our approach there are three tapes (modeled using a pair of transducers). The meaning representation is actually built within the finite-state framework. In Bellik's approach the meaning is built using operations attached to arcs in the ATN and is not modelled in finite-state terms.

10.1 Active and passive co-reference

Bellik (1997) distinguishes active and passive co-references between the modes in multimodal interfaces. *Active co-reference* involves combination of two deliberate input events. For example, the clicking on a window and saying *"close"*. In *passive co-reference*, an input event cannot be properly interpreted without knowing the state of another device. For example, the user says *"close"* while looking at a window on the display. The multimodal integration framework described here is primarily directed towards analysis and combination of what Bellik terms *active co-reference*. It is used for the combination of deliberate isolatable input events in multiple modes.

https://doi.org/10.1017/S1351324904003572 Published online by Cambridge University Press

One way to extend the approach to passive co-references would be to treat changes in gaze as actions and encode these actions using a vocabulary of symbols. We think, however, that the more appropriate strategy is not to represent passive inputs such as gaze in the same way as active inputs such as pen gestures. We believe that the current gaze position of the user should be made available to a multimodal dialog manager (Johnston *et al.* 2002) which can exploit that information in arriving at an interpretation.

Bellik (1997) presents an interesting example of passive co-reference in which the user may be looking at one of four windows, and the user says *"close"*. In our approach the multimodal grammar would allow for at least two interpretations of *"close"*. First, a command which requires an active deictic gesture to provide the referent. Second, a command which is unimodal speech and requires the referent to be available from context. If the user does not make an active gesture then the second unimodal interpretation will be the only one available after the finite-state multimodal integration. This interpretation is then passed to the multimodal dialog manager. The dialog manager searches the dialog history for a referent for the command. If a referent is not available then the dialog manager checks the output of the gaze recognizer to determine the target of the user's gaze and if the target is an object of appropriate type the dialog manager inserts the passive referent in the meaning representation.

11 Conclusion

We have presented here a novel approach to multimodal language processing in which spoken language and gesture are parsed and integrated by a single weighted finite-state device. This device provides language models for speech and gesture recognition and composes content from speech and gesture into a single semantic representation. Our approach is novel not just in addressing multimodal language but also in the encoding of semantics as well as syntax in a finite-state device.

Compared to previous approaches (Johnston *et al.* 1997; Johnston 1998a; Wu *et al.* 1999) which compose elements from *N*-best lists of recognition results, our approach provides an unprecedented potential for mutual compensation among the input modes. It enables gestural input to dynamically alter the language model used for speech recognition. Furthermore, our approach avoids the computational complexity of multidimensional multimodal parsing and our system of weighted finite-state transducers provides a well understood probabilistic framework for combining the probability distributions associated with speech and gesture input and selecting among multiple competing multimodal interpretations. Since the finite-state approach is more lightweight in computational needs, it can more readily be deployed on a broader range of platforms.

In ongoing research, we are collecting a corpus of multimodal data in order to formally evaluate the effectiveness of our approach and to train weights for the multimodal finite-state transducers. While we have concentrated here on understanding, in principle the same device could be applied to multimodal generation which we are currently investigating.

https://doi.org/10.1017/S1351324904003572 Published online by Cambridge University Press

We are also exploring techniques to extend compilation from feature structures grammars to FSTs (Johnson 1998) to multimodal unification-based grammars (Johnston 1998b).

Acknowledgements

We are thankful to Richard Cox, Mazin Rahim, and Candy Kamm for supporting this research effort. We also thank the reviewers for their insightful comments and suggestions which have improved the readability of the paper significantly. We would also like to thank Elliot Pinson, Marilyn Walker, Giuseppe Di Fabbrizio, Paolo Ruscitti and our experimental subjects.

References

Abney, S. (1991) Parsing by chunks. In: Berwick, R., Abney, S. and Tenny, C., editors, *Principle-based Parsing*, pp. 257–278. Kluwer.

André, E. (2002) Natural language in multimedia/multimodal systems. In: Mitkov, R., editor, *Handbook of Computational Linguistics*, pp. 650–669. OUP.

Bangalore, S. and J. ohnston, M. (2000) Tight-coupling of multimodal language processing with speech recognition. *Proceedings of ICSLP*, pp. 126–129. Beijing, China.

Bangalore, S. (1997) *Complexity of Lexical Descriptions and its Relevance to Partial Parsing*. PhD thesis, University of Pennsylvania, Philadelphia, PA.

Bangalore, S. and Riccardi, G. (2000) Stochastic finite-state models for spoken language machine translation. *Proceedings Workshop on Embedded Machine Translation Systems*, pp. 52–59.

Bellik, Y. (1995) *Interface Multimodales: Concepts, Modèles et Architectures*. PhD thesis, Paris XI University, France.

Bellik, Y. (1997) Media integration in multimodal interfaces. *Proceedings of IEEE Workshop on Multimedia Signal Processing*, Princeton, NJ.

Bellik, Y. and Burger, D. (1994) Multimodal interfaces: New solutions to the problem of computer accessibility for the blind. *Proceedings CHI 94*, pp. 24–28. Boston, MA.

Bellik, Y. and Teil, D. (1993) A multimodal dialogue controller for multimodal user interface management system, application: a multimodal window manager. *Proceedings INTERCHI 93*, pp. 24–29. Amsterdam, The Netherlands.

Bird, S. and Ellison, T. M. (1994) One level phonology: autosegmental representations and rules as finite automata. *Computational Linguistics*, **20**: 55–90.

Bolt, R. A. (1980) "Put-That-There": voice and gesture at the graphics interface. *Computer Graphics*, **14**(3): 262–270.

Buntschuh, B., Kamm, C., DiFabbrizio, G., Abella, A., Mohri, M., Narayanan, S., Zeljkovic, I., Sharp, R. D., Wright, J., Marcus, S., Shaffer, J., Duncan, R. and Wilpon, J. G. (1998) VPQ: A spoken language interface to large scale directory information. *Proceedings of ICSLP*, Sydney, Australia.

Carbonell, J. (1970) AI in CAI: An artificial intelligence approach to computer assisted instruction. *IEEE Trans. Man-Machine Syst*. **11**(4): 190–202.

Carpenter, R. (1992) *The Logic of Typed Feature Structures*. Cambridge University Press, England.

Cheyer, A. and Julia, L. (1995) Multimodal maps: An agent based approach. *Proceedings International Conference on Cooperative Multimodal Communication*, pp. 103–114. Eindhoven, The Netherlands.

Cohen, P. R., Johnston, M., McGee, D., Oviatt, S. L., Pittman, J., Smith, I., Chen, L. and Clow, LJ. (1998) Multimodal interaction for distributed interactive simulation.

https://doi.org/10.1017/S1351324904003572 Published online by Cambridge University Press

In: Maybury, M. and Wahlster, W., editors, *Readings in Intelligent Interfaces*, pp. 562–571. Morgan Kaufmann.

Jacob, R. J. K. (1986) A specification language for direct-manipulation user interfaces. *ACM Trans. Graphics*, 283–317.

Johnson, M. (1998) Finite-state approximation of constraint-based grammars using left-corner grammar transforms. *Proceedings COLING-ACL*, pp. 619–623. Montreal, Canada.

Johnston, M. (2000) Deixis and conjunction in multimodal systems. *Proceedings COLING 2000*, pp. 362–368. Saarbrücken, Germany.

Johnston, M. and Bangalore, S. (2000) Finite-state multimodal parsing and understanding. *Proceedings COLING 2000*, pp. 369–373. Saarbrücken, Germany.

Johnston, M., Bangalore, S., Vasireddy, G., Stent, A., Ehlen, P., Walker, M., Whittaker, S. and Maloor, P. (2002) MATCH: An architecture for multimodal dialog systems. *Proceedings ACL-02*, pp. 376–383. Philadelphia.

Johnston, M. (1998a) Multimodal language processing. *Proceedings ICSLP*, Sydney, Australia.

Johnston, M. (1998b) Unification-based multimodal parsing. *Proceedings COLING-ACL*, pp. 624–630. Montreal, Canada.

Johnston, M., Cohen, P. R., McGee, D., Oviatt, S. L., Pittman, J. A. and Smith, I. (1997) Unification-based multimodal integration. *Proceedings 35th ACL*, pp. 281–288. Madrid, Spain.

Joshi, A. and Hopely, P. (1997) A parser from antiquity. *Natural Lang. Eng.* **2**(4).

Kaplan, R. M. and Kay, M. (1994) Regular models of phonological rule systems. *Computational Linguistics*, **20**(3): 331–378.

Kay, M. (1987) Non-concatenative finite-state morphology. *Proceedings Third Conference of the European Chapter of the Association for Computational Linguistics*, pp. 2–10.

Kiraz, G. and Grimley-Evans, E. (1998) Multi-Tape Automata for Speech and Language Systems: A Prolog Implementation. In: Wood, D. and Yu, S., editors, *Automata Implementation: Lecture Notes in Computer Science 1436*, pp. 87–103. Springer.

Kornai, A. (1995) *Formal Phonology*. Garland Publishing, New York.

Koskenniemi, K. K. (1984) *Two-level morphology: a general computation model for word-form recognition and production*. PhD thesis, University of Helsinki.

Mohri, M. (1997) Finite-state transducers in language and speech processing. *Computational Linguistics*, **23**(2): 269–312.

Mohri, M., Pereira, F. C. N. and Riley, M. (1998) *A rational design for a weighted finite-state transducer library: Lecture notes in Computer Science 1436*, pp. 144–158. Springer.

Neal, J. G. and Shapiro, S. C. (1991) Intelligent multi-media interface technology. In: Sullivan, J. W. and Tyler, S. W., editors, *Intelligent User Interfaces*, pp. 45–68. ACM Press.

Nigay, L. and Coutaz, J. (1993) A design space for multimodal systems: Concurrent processing and data fusion. *Proceedings INTERCHI '93*, pp. 172–178. ACM Press.

Oviatt, S. L. (1997) Multimodal interactive maps: Designing for human performance. *Human-Computer Interaction*, pp. 93–129.

Oviatt, S. L. (1999) Mutual disambiguation of recognition errors in a multimodal architecture. *CHI '99*, pp. 576–583. ACM Press.

Pereira, F. and Shieber, S. (1987) *Prolog and Natural Language Analysis*. CSLI Lecture Notes, Stanford University Press.

Pereira, F. C. N. and Riley, M. D. (1997) Speech recognition by composition of weighted finite automata. In: Roche, E. and Schabes, Y., editors, *Finite State Devices for Natural Language Processing*, pp. 431–456. MIT Press.

Pereira, F. C. N. and Wright, R. (1997) Finite-state approximation of phrase structure grammars. In: Roche, E. and Schabes, Y., editors, *Finite State Devices for Natural Language Processing*, pp. 149–173. MIT Press.

Quine, W. V. O. (1969) *Ontological Relativity and Other Essays*. Columbia University Press.

Riccardi, G., Pieraccini, R. and Bocchieri, E. (1996) Stochastic Automata for Language Modeling. *Computer Speech & Lang.* **10**(4): 265–293.

https://doi.org/10.1017/S1351324904003572 Published online by Cambridge University Press

Roche, E. (1999) Finite state transducers: parsing free and frozen sentences. In: Kornai, A., editor, *Extended Finite State Models of Language*, pp. 103–120. Cambridge University Press.

Rosenberg, A. L. (1964) On n-tape finite state acceptors. *FOCS*, pp. 76–81.

van Noord, G. (1997) FSA utilities: A toolbox to manipulate finite-state automata. In: Wood, D., Raymond, D. and Yu, S., editors, *Automata Implementation: Lecture Notes in Computer Science 1260*, pp. 87–108. Springer.

Vo, M. T. (1998) *A Framework and Toolkit for the Construction of Multimodal Learning Interfaces*. PhD thesis, Carnegie Mellon University.

Wahlster, W. (2002) SmartKom: Fusion and fission of speech, gestures, and facial expressions. *Proceedings 1st International Workshop on Man-Machine Symbiotic Systems*, pp. 213–225. Kyoto, Japan.

https://doi.org/10.1017/S1351324904003572 Published online by Cambridge University Press