Finite-state Methods for Multimodal Parsing and Integration

Michael Johnston
AT&T Labs - Research
Shannon Laboratory, 180 Park Ave
Florham Park, NJ 07932, USA
johnston@research.att.com

Srinivas Bangalore
AT&T Labs - Research
Shannon Laboratory, 180 Park Ave
Florham Park, NJ 07932, USA
srini@research.att.com

1 Introduction

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

In Johnston and Bangalore (2000) we showed how finite-state methods can be employed in a new and different task - parsing, integration, and understanding of multimodal input. Our approach addresses the particular case of multimodal input to a mobile device where the modes are speech and gestures made on the display with a pen, but has far broader application. The approach uses a multimodal grammar specification which is compiled into a finite-state device running on three tapes. This device takes as input a speech stream and a gesture stream and outputs their combined meaning. The approach overcomes the computational complexity of unification-based approaches to multimodal processing (Johnston, 1998), enables tighter coupling with speech recognition, and enables straightforward composition with other kinds of language processing such as finite-state translation (Bangalore and Riccardi, 2000).

In this paper, we present a revised and updated finite-state model for multimodal language processing which incorporates a number of significant advancements to our approach. We show how gesture symbols can be decomposed into attributes in order to reduce the alphabet of gesture symbols and enable underspecification of required gestures. We present a new mechanism for abstracting over gestural content that cannot be captured in the finite-state machine.[1] We address the problems relating to deictic numerals (Johnston, 2000) by introducing a new mechanism for aggregation of adjacent gestures. The examples we use are drawn from a new more sophisticated multimodal application which provides mobile access to city information such as the locations of restaurants and theatres (we will demonstrate this application as part of our presentation). We will also draw examples from other applications as needed. In addition to addressing multimodal rather than unimodal input, another novel aspect of our approach is that we used the finite-state representation to build the meaning representation.

We first present the basics of the finite-state approach and then go on to discuss each of the innovations in turn.

[1] This overcomes a number of difficulties with the previous buffering/variable mechanism we used.

2 Finite-state models for multimodal processing

Multimodal integration involves merging semantic content from multiple streams to build a joint interpretation for a multimodal utterance. We employ a finite-state device to parse multiple input streams 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 streams 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.

In the city information application example in this paper, users issue spoken commands such as `tell me about these two restaurants` while gesturing on icons on a dynamically generated map display, or `show cheap Italian restaurants in this neighborhood` while drawing an area on the display. The structure and interpretation of multimodal commands of this kind is captured declaratively in a multimodal context-free grammar. We present a fragment capable of handling such commands in Figure 1.

The non-terminals in the multimodal grammar are atomic symbols. The multimodal aspects of the grammar become apparent in the terminals. Each terminal 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 (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. Sequences of symbols such as *G area location* indicate the presence of a particular kind of gesture in the gesture stream, while those like *SEM* are used as references to entities referred to by the gesture. In our previous application, the semantic representation consisted of predicates. In the current application we are experimenting with the use of XML as the meaning representation. The meaning tape contains symbols which when concatenated together form coherent XML expressions.

Our approach makes certain simplifying assumptions with respect to temporal constraints. In multi-gesture utterances the primary function of temporal constraints is to force an order on the gestures. If you say `move this here` and make two gestures, the first gesture corresponds to `this` and the second gesture to `here`. Our multimodal grammars encode order but do not impose explicit temporal constraints. However, general temporal constraints between speech and the first gesture can be

S	→	COMMAND
COMMAND	→	show:eps:<show> NP eps:eps:</show>
COMMAND	→	tell:eps:<info> me:eps:eps about:eps:eps DEICTICNP eps:eps:</info>
NP	→	eps:eps:<restaurant> CUISMOD restaurants:eps:eps LOCMOD eps:eps:</restaurant>
DEICTICNP	→	DDETSG SELECTION eps:1:eps RESTSG eps:eps:<restaurant> Entry eps:eps:</restaurant>
DEICTICNP	→	DDETPL SELECTION NUM RESTPL eps:eps:<restaurant> Entry eps:eps:</restaurant>
SELECTION	→	eps:area:eps eps:selection:eps
CUISMOD	→	eps:eps:<cuisine> CUISINE eps:eps:</cuisine>
CUISINE	→	italian:eps:italian
CUISINE	→	chinese:eps:chinese
LOCMOD	→	eps:eps:<location> LOCATION eps:eps:</location>
LOCMOD	→	eps:eps:eps
LOCATION	→	in:eps:eps this:G:eps area:area:eps eps:location:eps Entry
LOCATION	→	along:eps:eps this:G:eps route:line:eps eps:location:eps Entry
DDETSG	→	this:G:eps
DDETPL	→	these:G:eps
NUM	→	two:2:eps
NUM	→	three:3:eps
RESTSG	→	restaurant:restaurant:eps
RESTPL	→	restaurants:restaurant:eps
Entry	→	eps:SEM:SEM

Figure 1: Multimodal grammar fragment

enforced before the FSA is applied.

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 $(W \cup \{eps\}) : (G \cup \{eps\}) : M^*$ where W is the vocabulary of speech, G is the vocabulary of gesture=$GestureSymbols \cup EventSymbols$; $GestureSymbols$ ={G, area, location, restaurant, 1, ...} and an event symbol $EventSymbol$ ={SEM}. M is the vocabulary to represent meaning and includes the event symbol ($EventSymbols \subset M$).

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 1 translates into the FSA in Figure 2, a three-tape finite state device capable of composing two input streams into a single output semantic representation stream.

While a three-tape finite-state automaton is feasible in principle (Rosenberg, 1964), currently available tools for finite-state language processing (Mohri et al., 1998; van Noord, 1997) 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. In order to implement our approach, we convert the three-tape FSA (Figure 2) into an FST, by decomposing the transition symbols into an input component ($G \times W$) and output component M, thus resulting in a function, $\mathcal{T}$:$(G \times W) \rightarrow M$. This corresponds to a transducer in which gesture symbols and words are on the input tape and the meaning is on the output tape. The domain of this function $\mathcal{T}$ can be further curried to result in a transducer that maps $\mathcal{R}$:$G \rightarrow W$. 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 sections we discuss several advancements in our finite-state approach to understanding multimodal input.

3 Gesture symbol complexes

In our original approach, atomic symbols were used in the multimodal grammar and corresponding machine to represent different types of gestures. For example, *Gp* was used for gestural reference to a person, *Go* for a reference to an organization, *2Gp* for a gestural reference to a set of two people and so on. In our current application we are exploring the idea of decomposing the gesture symbols into sequences of symbols each of which conveys a specific attribute of the content such as type or number. This facilitates reference to sets of specific symbols. It also limits the number of symbols that are needed for gesture. It plays an important role in enabling the storage of specific gesture contents (See Section 4) and aggregation (See Section 5). The gesture symbol complexes follow a basic form: *G FORM MEANING (NUMBER TYPE) SEM*. *FORM* indicates the physical form of the gesture: and has values such as area, point, line, arrow. *MEANING* indicates the specific meaning of that form for example an *area* can be either a *location* or a *selection*. *NUMBER* and *TYPE* are only found with *selection*. They indicate the number of entities selected (*1,2,3, many*) and the specific type of entity (*restaurant, theatre*). The *TYPE mixed* is used for gestures at collections of entities of varied different types. In order to facilitate abstraction and recomposition of specific gestural content (see Section 4), the specific content is mapped to a distinguished symbol *SEM*, while the other attributes of the gesture are mapped to themselves.

As an example, if the user draws an area on the screen which contains two restaurants, which have identifiers *id1* and *id2*, the resulting gesture lattice will be as in Figure 3. If the speech is `show me chinese restaurants in this neighborhood` then the first path will be chosen when the multimodal finite-state device is applied. If the speech is `tell me about these two restaurants` then the second, *selection*, path will be chosen.

If instead the user circles a restaurant and a theatre the lattice would be as in Figure 4 and if they say `tell me about this theatre` the third path will be taken. But if they say `tell me about these two` the fourth path will be taken. This approach allows

for cases where a user circles several entities and selects a specific one by type.

If we did not split the symbols we would need a *G_area_location* symbol, *G_area_selection_1_restaurant*, *G_area_selection_2_restaurant* etc., significantly increasing the alphabet of gesture symbols. The split symbols also allow more perspicuous representation of more general categories. For example, if `place` in `tell me about this place` can refer to either a restaurant or a theatre then it can be assigned both arcs in the lattice. As shown in the next two sections, the splitting of gestures symbols also plays an important role in the abstraction and recovery of specific gestural content, and the process of aggregation.

4 Recovering gesture content by composition

In order to capture multimodal integration using finite-state methods, it is necessary to abstract over certain aspects of the gestural content. For example, it is not possible to capture all of different possible sequences of coordinates that occur in gesture so that they can be copied from the gesture input tape to the meaning output tape. In our previous approach we assigned the specific content of gestures to a series of numbered variables *e1, e2, e3 . .* There are a number of limitations to this approach. The number of gestural inputs that can be handled is limited by the number of variables used. If a large number are used then the resulting multimodal finite-state device increases significantly in size since an arc is needed for each variable in every place where gesture content can be copied onto the meaning tape. This becomes a significant problem when aggregation of gestures is considered (See Section 5 below) since a new variable is needed for each combination of gestures. We have developed a new approach to this problem. The gestural input is represented as a transducer that maps $\mathcal{S} : I \rightarrow G$ where G are gesture symbols and I are the specific interpretations (see Figure 4). The G side contains indication of the type of gesture and its properties. In any place where there is specific content such as a list of entities or points in the gesture stream the symbol in G is the reserved symbol SEM. The specific content is placed on the I side opposite SEM. All G symbols other than SEM match with an identical symbol on I. In order to carry out the multimodal composition with the transducer $\mathcal{R} : G \rightarrow W$ and $\mathcal{T} : (G \times W) \rightarrow M$, the output projection G of $\mathcal{S} : I \rightarrow G$ is taken. After composition we take a projection $\mathcal{U} : G \rightarrow M$ of the resulting *G_W:M* machine, basically we factor out the speech *W* information. We then compose $\mathcal{S}$ and $\mathcal{U}$ yielding *I:M*. In order to read off the meaning we concatenate symbols from the *M* side. If the *M* symbol is SEM we instead take the *I* symbol for that arc. As an example, when the user says `show chinese restaurants in this area` and the gesture is as in Figure 3 the resulting *G_W:M* machine is as in Figure 6. In order to compose this with Figure 3 the *_W* is removed. Reading of the meaning from the resulting *I:M* we have the representation in Figure 7.

5 Aggregation for Deictic Numerals

Johnston (2000) examines the problems posed for the unification-based approach to multimodal parsing (Johnston, 1998) by deictic numeral expressions such as `these four restaurants`. The problem is that there are a multitude of different possible sequences of gestures that are compatible and should be integrated with this spoken phrase. The user might circle a set of four restaurants, they might circle four individual restaurants, they might circle two sets of two, circle one and circle three etc. Capturing all of these possibilities in the spoken language grammar significantly increases its size and complexity and any plural expression is made massively ambiguous. The suggested alternative in (Johnston, 2000) is to have the deictic numeral subcategorize for a plurality of the appropriate number and predictively apply a set of gesture combination rules in order combine elements of gestural input into the appropriate pluralities. In the finite-state approach this is achieved using a method we term *aggregation* which serves a pre-processing phase on the gesture input lattice. Additional branches are added to the lattice which represent combinations of adjacent elements. This process needs to combine the specific semantic contents (*SEM* values) and so is carried on outside of the finite-state representation. An expression such as `these three restaurants` is assigned the gesture stream *G 3 restaurant SEM* by the multimodal grammar.[2] This will directly combine with a single area gesture containing three restaurants. In order to combine with a sequence of three separate gestures on single restaurants aggregation must apply. The gesture lattice for three gestures in a row is *G:G 1:1 restaurant:restaurant ([id1]):SEM G:G 1:1 restaurant:restaurant ([id2]):SEM G:G 1:1 restaurant:restaurant ([id3]):SEM*. In pre-processing the gesture lattice is parsed and adjacent selections of identical type are composed and new branches added to the lattice. In this case, three more gestures are added to the lattice as in Figure 5. This can be thought of as a closure on the gesture lattice of a function which combines adjacent gestures of identical type.

When this gesture lattice is combined with the spoken input by the multimodal finite state device the speech will pick out the *G 3 restaurant* path in the gesture lattice. We term this kind of aggregation *type specific*. We also use *type non-specific* aggregation to generate aggregates of mixed type. For example in the case where the user says `tell me about these two` and circles a restaurant and then a theatre, *non-type specific* aggregation applies to combine the two gestures into an aggregate of mixed type *G:G 2:2 mixed:mixed ([id4,id5]):SEM* and this is able to combine with `these two`. In order to preserve an indication of the original sequence of gestures the newly added paths can be assigned lower weights.

[2] We are simplifying the gesture symbol complex here for ease of exposition. In each case, the *area selection* would come between the *G* and the number.

6 Finite-state Semantic Representation

A significant aspect of our approach is that in addition to capturing the structure of the input we also build the semantic representation within the finite-state framework. In our original application (Johnston and Bangalore 2000), we generated a prolog-like logical representation, for example: *email([person(id1),organization(id2)])*. In our city information application we are exploring the use of an XML-based meaning representation. The symbols on the meaning tape can be concatenated together to form coherent XML expressions which are then evaluated by the underlying application. For example, `show chinese restaurants in this neighborhood` yields Figure 7.

7 Conclusion

We have presented an approach to multimodal language understanding in which speech and gesture inputs are assigned a combined meaning by a finite-state device. This work is novel not just in its application to multimodal input but also in that it assigns semantic representation using a finite-state device. We have highlighted a number of recent advancements in the approach since Johnston and Bangalore 2000. Gesture symbols have been split up into gesture symbol complexes which allow for a broader range of expression. The specifics of gestural input are abstracted over into a gesture lattice and reinserted into the selected multimodal interpretation without the use of variables or buffers. We have addressed the problem of deictic numerals brought up by Johnston 2000 using an aggregation mechanism which acts as a pre-processing stage on the gesture input and have explored the use of XML as an output meaning representation. In ongoing work we are exploring methods for automatically generating the multimodal grammar from a feature-based representation, and examining how the machine can be used in reverse for multimodal generation tasks.

References

Steven Abney. 1991. Parsing by chunks. In Robert Berwick, Steven Abney, and Carol Tenny, editors, *Principle-based parsing*. Kluwer Academic Publishers.

Srinivas Bangalore and Giuseppe Riccardi. 2000. Stochastic finite-state models for spoken language machine translation. In *Proceedings of the Workshop on Embedded Machine Translation Systems*.

Srinivas Bangalore. 1997. *Complexity of Lexical Descriptions and its Relevance to Partial Parsing*. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA, August.

Michael Johnston and Srinivas Bangalore. 2000. Finite-state multimodal parsing and understanding. In *Proceedings of COLING 2000*, Saarbruecken, Germany.

Michael Johnston. 1998. Unification-based multimodal parsing. In *Proceedings of COLING-ACL*, pages 624–630, Montreal, Canada.

Michael Johnston. 2000. Deixis and conjunction in multimodal systems. In *Proceedings of COLING 2000*, Saarbruecken, Germany.

Aravind Joshi and Philip Hopely. 1997. A parser from antiquity. *Natural Language Engineering*, 2(4).

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

Lauri Kartunnen. 1991. Finite-state constraints. In *Proceedings of the International Conference on Current Issues in Computational Linguistics*, Universiti Sains Malaysia, Penang, Malaysia.

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

Mehryar Mohri, Fernando C. N. Pereira, and Michael Riley. 1998. *A rational design for a weighted finite-state transducer library*. Number 1436 in Lecture notes in computer science. Springer, Berlin ; New York.

Mark-Jan Nederhof. 1997. Regular approximations of cfls: A grammatical view. In *Proceedings of the International Workshop on Parsing Technology*, Boston.

Fernando C.N. Pereira and Michael D. Riley. 1997. Speech recognition by composition of weighted finite automata. In E. Roche and Schabes Y., editors, *Finite State Devices for Natural Language Processing*, pages 431–456. MIT Press, Cambridge, Massachusetts.

Fernando C.N. Pereira and R. Wright. 1997. Finite-state approximation of phrase structure grammars. In E. Roche and Schabes Y., editors, *Finite State Devices for Natural Language Processing*. MIT Press, Cambridge, Massachusetts.

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

Emmanuel Roche. 1999. Finite state transducers: parsing free and frozen sentences. In András Kornai, editor, *Extended Finite State Models of Language*. Cambridge University Press.

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

Gertjan van Noord. 1997. FSA Utilities: A Toolbox to Manipulate Finite-state Automata. In Derick Wood Darrell Raymond and Sheng Yu, editors, *Automata Implementation. Lecture Notes in Computer Science 1260*. Springer Verlag.

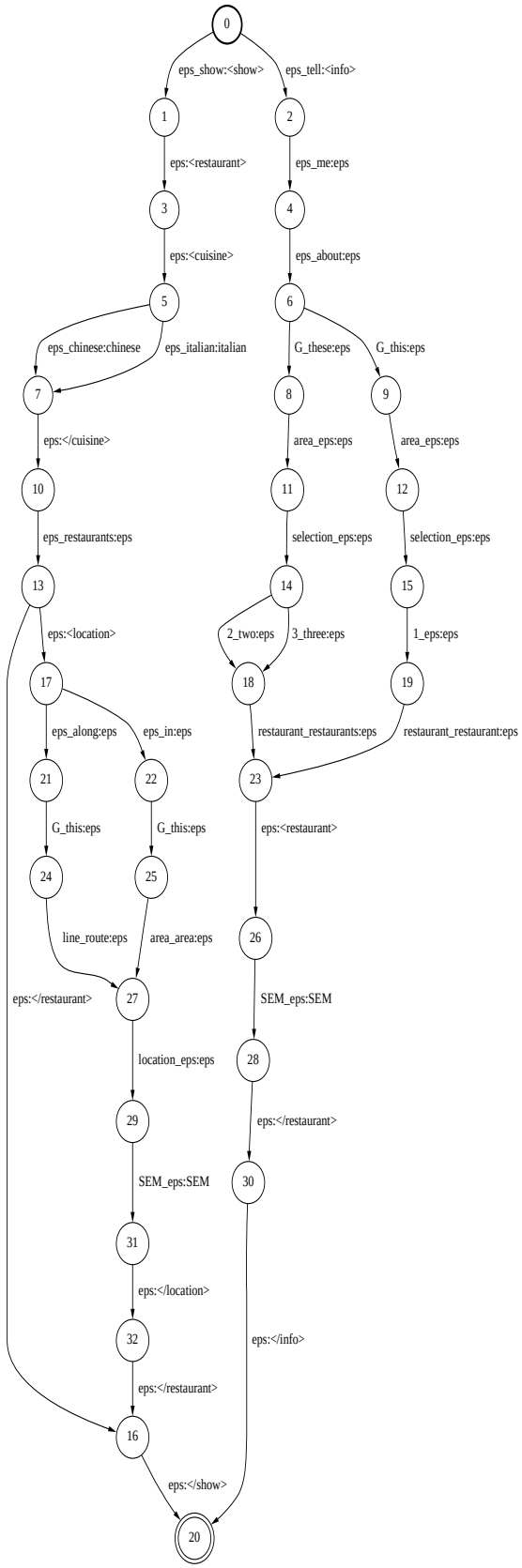

Figure 2: Multimodal three-tape FSA

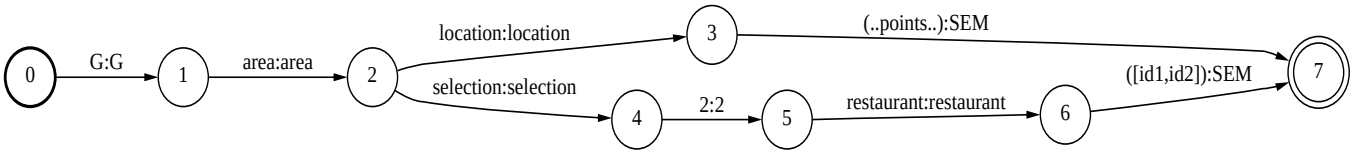

Figure 3: Gesture Lattice: Two restaurants

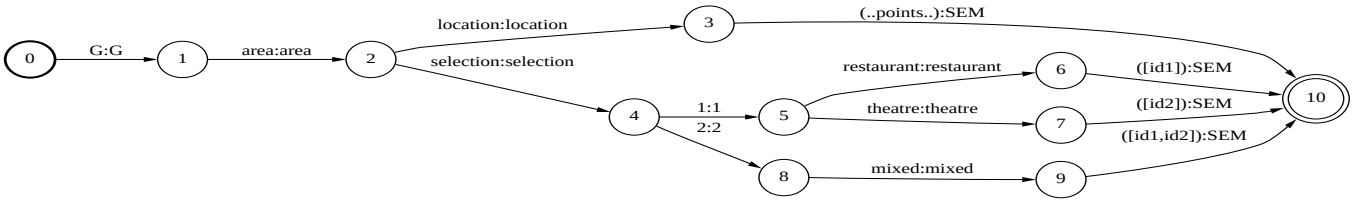

Figure 4: Gesture Lattice: Restaurant and theatre

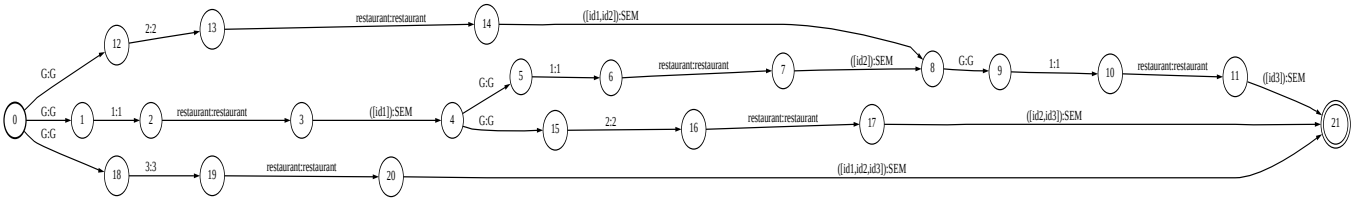

Figure 5: Aggregated Lattice

Figure 6: G_W:M machine

```
<show> <restaurant>
<cuisine>chinese</cuisine>
<location>(...points...)</location>
</restaurant> </show>
```

Figure 7: Sample XML output