Almost Flat Functional Semantics for Speech Translation

Manny Rayner[1], **Pierrette Bouillon**[1], **Beth Ann Hockey**[2], **Yukie Nakao**[3]

[1] **University of Geneva, TIM/ISSCO**
40 bvd du Pont-d'Arve
CH-1211 Geneva 4, Switzerland
Emmanuel.Rayner@issco.unige.ch
Pierrette.Bouillon@issco.unige.ch

[2] **UCSC UARC, Mail Stop 19-26**
NASA Ames Research Center
Moffet Field, CA 94035
bahockey@ucsc.edu

[3] **University of Nantes, LINA**
2, rue de la Houssinière
BP 92208 44322 Nantes Cedex 03
yukie.nakao@univ-nantes.fr

Abstract

We introduce a novel semantic representation formalism, Almost Flat Functional semantics (AFF), which is designed as an intelligent compromise between linguistically motivated predicate/argument semantics and *ad hoc* engineering solutions based on flat feature/value lists; the central idea is to tag each semantic element with the functional marking which most closely surrounds it. We argue that AFF is well-suited for medium-vocabulary speech translation applications, and describe simple and general algorithms for parsing, generating and performing transfer using AFF representations. The formalism has been fully implemented within a medium-vocabulary interlingua-based Open Source speech translation system which translates between English, French, Japanese and Arabic.

1 Introduction

Many speech translation architectures require some way to represent the meaning of spoken utterances, but even a brief review of the literature reveals a serious divergence of opinion as to how this may best be done. At risk of oversimplifying a little, there are two competing heritages. On the one hand, there is the mainstream computational semantics approach, which ultimately goes back to philosophers like Montague, Russell and Frege and views predicate calculus as the paradigm representation language. On this view of things, a suitable way to represent meaning is to use complex structures, in which components and relationships are based on deep grammatical functions. Typical ways to realise this strategy are unscoped logical forms, neo-Davidsonian semantics, minimal recursion semantics, and similar formalisms. Thus a sentence like "I want a pepperoni pizza" might be represented as something like

```
want1(E, X, Y),
ref(X, pronoun(i)),
quant(Y, indef), pizza1(Y),
pepperoni1(Z), nn(Z, Y)
```

Approaches based in the linguistic tradition were dominant about 10 to 15 years ago, when they were used in major systems like Germany's Verbmobil (Wahlster, 2000) and SRI's Spoken Language Translator (Rayner et al., 2000). They are still reasonably popular today, as exemplified by major systems like PARC's XLE (Riezler et al., 2002).

The competing heritage has its roots in engineering approaches to spoken language systems, which historically have been intimately connected with Machine Learning. On this view of things, a typical semantic representation is a flat list of feature-value pairs, with the features representing semantic concepts: here, "I want a pepperoni pizza" would be represented as something like

```
[utterance_type=request,
 food=pizza, type=pepperoni]
```

It is interesting to see how little contact there has been between these two traditions. Writers on formal semantics usually treat *ad hoc* feature-value representations as not even worthy of serious discussion. Conversely, proponents of engineering/machine learning approaches often assume in practice that all semantic representations will be some version of a flat feature-value list; a good

Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008), pages 713–720
Manchester, August 2008

example of this tendency is Young's widely cited 2002 survey of machine learning approaches to spoken dialogue (Young, 2002).

Trying to be as neutral as possible, it is reasonable to argue that both approaches have important things to offer, and that it is worth trying to find some compromise between them. Other things being equal, flat feature-value representations have desirable formal properties: they are simple, and easy to manipulate and reason with. Their drawback is that they are an impoverished representation language, which can lose important information. This means that concepts may be impossible to represent, or, alternately viewed, that the representation format may conflate concepts which we would prefer to distinguish. In the other direction, hierarchical logic-based representations are highly expressive, but pose much more serious challenges in terms of formal manipulability. Although they are more easily capable of representing semantic distinctions, it is harder to use them to perform concrete reasoning operations. In translation systems, these abstract issues manifest themselves in a tradeoff between complexity of translation rules, and ambiguity of semantic representations. A flat semantic representation formalism means that translation rules are simple to write; however, it also means that the semantic representations they operate on are more likely to be ambiguous.

In this paper, we will explore the tradeoffs between the two competing positions outlined above in the context of a concrete Open Source system, the MedSLT medical speech translator. Previous versions of MedSLT have used a representation strategy intermediate between the "logic-based semantics" and the "flat semantics" approaches, though much closer to the "flat" end of the scale. We will discuss the strengths and weaknesses of the original MedSLT representation formalism, and then present a revised version, "Almost Flat Functional Semantics" (AFF). As the name suggests, AFF incorporates functional markings, characteristic of a logic-based semantics approach, into a representation formalism which still mainly consists of flat list structures. We will show how grammars using semantics written in AFF can be compiled into parsers and generators, and describe a simple formalism that can be used to specify rules for translating AFF expressions into AFF expressions. Finally, we will show how use of AFF in MedSLT has allowed us to address in a principled way most of the examples which are problematic for the original version of the system, while still retaining a simple and transparent framework for writing translation rules.

2 The MedSLT System

MedSLT (Bouillon et al., 2005) is a medium-vocabulary Open Source speech translation system for medical domains, implemented using the Open Source Regulus compiler (Rayner et al., 2006) and the Nuance recognition platform. Processing is primarily rule-based. Recognition uses a grammar-based language model, which produces a source-langage semantic representation. This is first translated by one set of rules into an interlingual form, and then by a second set into a target language representation. A target-language grammar, compiled into generation form, turns this into one or more possible surface strings, after which a set of generation preferences picks one out. Finally, the selected string is realised in spoken form. There is also some use of corpus-based statistical methods, both to tune the language model (Rayner et al., 2006, Section 11.5) and to drive a robust embedded help system (Chatzichrisafis et al., 2006).

The treatment of syntactic structure is a carefully thought-out compromise between linguistic and engineering traditions. All grammars used are extracted from general linguistically motivated resource grammars, using corpus-based methods driven by small sets of examples (Rayner et al., 2006, Chapter 9). This results in a simpler and flatter grammar specific to the domain, whose structure is similar to the *ad hoc* phrasal grammars typical of engineering approaches. The treatment of semantics is however less sophisticated, and basically represents a minimal approach in the engineering tradition. Each lexicon item contributes a set of zero or more feature-value pairs (in most cases exactly one pair). Most of the grammar rules simply concatenate the sets of pairs received from their daughters. A small number of rules, primarily those for subordinate clauses, create a nested substructure representing the embedded clause. Figure 1 shows an example representation.

It should be obvious from the example that the flat representation is potentially very ambiguous, since nearly all information about grammatical functions has been lost. The example also illustrates, however, why this is often unimportant in

```
[[utterance_type,sentence],
 [pronoun,vous],
 [path_proc,avoir],
 [voice,active],
 [tense,present],
 [cause,nausée], [sc,quand],
 [clause,
   [[pronoun,vous],
    [symptom,mal],
    [path_proc,avoir],
    [voice,active],
    [tense,present]]])]
```

Figure 1: Semantic representation produced by the current MedSLT system for the French sentence *Avez-vous des nausées quand vous avez mal?* (“Do you have nausea when you have the pain?”)

practice. From a purely syntactic point of view, the fragment

```
[[pronoun,vous],
 [path_proc,avoir],
 [cause,nausée]]
```

could either represent *vous avez des nausées* (“you have nausea”) or *des nausées vous ont* (“nausea has you”). Except, possibly, in certain kinds of literary contexts, the second realisation is so implausible that it can be discounted. It is thus reasonable to add sortal constraints to the lexical entries involved, which permit *des nausées* to occur in well-formed utterances as the object of *avoir*, but not as its subject. Thus the representation is in fact unambiguous, and will only generate one surface realisation.

With the moderate vocabularies used by MedSLT (for example, the current French module has a vocabulary of about 1 100 surface words), the vast majority of constructions can be rendered unambiguous using similar strategies. The result is that most translation rules are easy to write, since they have to do no more than map lists of feature-value pairs to lists of feature-value pairs. To take a typical example, the Japanese question *itami wa koutoubu desu ka* (“pain-TOPIC back-part-head is-Q”) receives the representation

```
[[utterance_type,sentence],
 [symptom,itami],
 [body_part,koutoubu],
 [verb,desu], [tense,present]]
```

which we wish to map to the interlingua representation

```
[[utterance_type,ynq], [verb,be],
 [tense,present], [voice,active],
 [symptom,pain], [prep,in_loc],
 [part,back], [body_part,head]]
```

(“is the pain in the back of the head”). In a more expressive semantic framework, the structural mismatches here would be non-trivial to resolve. In the flat MedSLT notation, we only need the following two list-to-list translation rules:

```
transfer_rule(
  [[body_part,koutoubu]],
  [[prep,in_loc], [part,back],
   [body_part,head]]).
```

(*koutoubu* → “in the back of the head”) and

```
transfer_rule(
  [[verb, desu]],
  [[verb, be]]).
```

(*desu* → “is”).

As usual, however, we pay a price for simplicity. In the terminology of Statistical Machine Translation, what we are essentially doing here is weakening the channel model, and relying on the strength of the target language model. This is a reasonable strategy partly because of the restricted nature of the domain, and partly because of the fact that the initial parsing stage makes it possible for us to work with bags of concepts rather than bags of words; clearly, bags of concepts are more expressive.

None the less, it is normal to expect the underspecified channel model to cause some problems, and this indeed proves to be the case. Although most semantic relationships in the domain are unambiguous even as bags of concepts (“back of the head” is possible; “head of the back” isn’t), there are unpleasant counterexamples. For instance, {“visit”, “doctor”, “patient”} can be realised as either “patient visits doctor” or “doctor visits patient”. Similarly, {“precede”, “nausea”, “headache”} can be either “nausea precedes headache” or “headache precedes nausea”. Cases like these must be dealt with using *ad hoc* solutions based on domain pragmatics. In the current version of the system, “patient visits doctor” is forced by producing both surface realisations, and defining a generation preference. In the case of {“precede”, “nausea”, “headache”}, the problem is addressed by dividing symptoms into “primary” (the symptom the patient is being examined for, e.g. “headache”) and “secondary” (other possibly

```
utterance:[sem=concat(Verb, [[tag, obj, Np]])] -->
    verb:[sem=Verb], np:[sem=Np].

np:[sem=concat(Adj, Noun)] -->
    spec:[], ?adj:[sem=Adj], noun:[sem=Noun].

np:[sem=concat(Np, PP)] -->
    np:[sem=Np], pp:[sem=PP].

pp:[sem=[[tag, Tag, Np]]] -->
    prep:[sem=Tag], np:[sem=Np].

verb:[sem=[[action, grasp]]] --> grasp.
noun:[sem=[[thing, block]]] --> block.
noun:[sem=[[loc, table]]] --> table.
adj:[sem=[[colour, red]]] --> red.
spec:[] --> the.
prep:[sem=on] --> on.
```

Figure 2: Toy grammar with nested predicate-argument semantics.

related symptoms, e.g. "nausea"). It is reasonable in practice to assume that the doctor will only be interested in secondary symptoms that may cause primary ones, and hence will precede them.

Although each language in the current version of MedSLT only contains a handful of similar cases, solutions like those outlined above are both inelegant and brittle. It would be desirable to find some more principled way to deal with them; we would, however, like to do this without sacrificing the appealing simplicity of the translation rule formalism. In the next section, we will show how it is possible to reconcile these two conflicting goals.

3 Almost Flat Functional Semantics

As we have seen, the problem with a simple bag-of-concepts representation is its ambiguity; what we would like to do is find some principled way to reduce that ambiguity, without greatly increasing the formalism's representational complexity. At this point, a linguistic intuition is helpful. The bag-of-concepts representation can reasonably be thought of as an artificial free word-order language. There are many natural free word-order languages; the reason why they are in general no more ambiguous than fixed word-order languages is that they use case-marking to convey functional information which constrains the space of possible interpretations. For speakers of European languages, the best-known example will probably be Classical Latin. For instance, when St. Jerome wrote *Amor ordinem nescit* ("love-NOM order-ACC not-know-PRES-3-SING"), the case-markings make it clear that he meant "Love does not know order" rather than "Order does not know love".

The comparison with free word-order languages suggests a natural extension of the original bag-of-concepts representation, where each element is associated with an additional functional tag which does the work that a case-marking would do in a natural free word-order language. It also suggests a simple construction which can be used to create an unordered linear representation that includes functional tags. We start by defining a standard nested predicate-argument semantics; we then flatten the representation of each clause S, marking each primitive semantic element with the immediately surrounding functional tag in S, or with a null marking if there is no such tag. The resulting semantic representations still represent each clause as an unordered list, but in contrast to the MedSLT bag-of-concepts representation now include functional information. We will call this style of representation *Almost Flat Functional* (AFF) semantics; the "almost" comes from the fact that there is still a minimal amount of nested structure, representing the distinction between main and embedded clauses.

Figures 2 and 3 give a concrete illustration of the

```
[[action, grasp],
 [tag, obj,
   [[colour, red],
    [thing, block],
    [tag, on,
      [[loc, table]]]]]]
```

```
[null=[action, grasp],
 obj=[colour, red],
 obj=[thing, block],
 on=[loc, table]]
```

Figure 3: Construction of AFF representation for "grasp the red block on the table". The AFF representation (right) is a flattened version of the original nested predicate-argument one (left).

AFF construction. Figure 2 presents a toy Regulus grammar, which allows a few sentences like "grasp the red block on the table" and assigns a nested functional semantics to them. The representations of most constituents are unordered lists. In the case of `utterance` and `np`, these are formed by concatenating the representations of their daughters. There are two examples of functional markings: the rule for `utterance` wraps an `[tag, obj ...]` around its `np` daughter, and the rule for `pp` wraps a tag around its `np` daughter, whose label is determined by the semantic value of the `p` daughter.

Figure 3 introduces the AFF construction itself. The left-hand side of the figure shows the nested predicate-argument representation of the sentence, in which elements of the form

```
[tag, Tag, Arg]
```

represent tags and their associated arguments. The right-hand side shows the derived AFF representation, where each element that is within the scope of a `[tag ...]` has been marked with the tag that would be immediately above it in the nested version. Thus the element `[loc, table]` is inside the scope of both the `obj` tag and the `on` tag; however, the AFF version assigns it the `on` tag, since this is the innermost one.

In the rest of this section, we will describe how we can parse surface strings into AFF representations, generate surface strings from AFF representations, and define translation rules which map AFF representations to AFF representations.

3.1 Analysis and generation

For both analysis and generation, the starting point is a grammar with a nested predicate-argument semantics like the one shown in the left half of Figure 3. Analysis is straightforward. We first use a standard parser-generator to compile the grammar into a parser; the nested predicate-argument representations it produces are then subjected to a post-processing phase, which flattens them in the way illustrated in the figure.

This simple approach is however not feasible for generation, since the flattening operation is highly non-deterministic in the reverse direction; finding all possible "unflattenings" and then attempting to generate from each one would in most cases be hopelessly inefficient. A better solution is to transform the original grammar into one with AFF semantics, where the current functional marking is specified as an extra features on relevant constituents, and percolated through the rules. In effect, the "unflattening" and generation operations can now proceed simultaneously, with each one constraining the other.

Figure 4 presents an example, showing the result of performing this transformation on the toy Regulus grammar from Figure 2. Here, the original `[tag, ...]` wrappers have been removed, and replaced by the new feature `tag`, which has been added to all constituents whose semantics is a list of items of the form `Tag=Value`. The value of the `tag` feature on each constituent where it is defined is the tag for the most closely enclosing `[tag, ...]` in the original grammar; these values are percolated down to the lexical rules, where they unify with the tags on the semantic fragment contributed by the rule. The transformation is straightforward to define in its general form, and the transformed grammars can be readily compiled into efficient generators by standard feature-grammar generator-compiler algorithms like Semantic Head-Driven Generation (Shieber et al., 1990). For the concrete experiments described later, we used a slightly extended version of the Open Source Regulus generator compiler.

3.2 Transfer

Our basic strategy for defining transfer between AFF expressions is to make it as close as possible to transfer on the original bag-of-concepts

```
utterance:[sem=concat(Verb, Np)] -->
    verb:[sem=Verb, tag=null], np:[sem=Np, tag=obj].

np:[sem=concat(Adj, Noun), tag=Tag] -->
    spec:[], ?adj:[sem=Adj, tag=Tag], noun:[sem=Noun, tag=Tag].

np:[sem=concat(Np, PP), tag=Tag] -->
    np:[sem=Np, tag=Tag], pp:[sem=PP, tag=Tag].

pp:[sem=Np] -->
    prep:[sem=Tag], np:[sem=Np, tag=Tag].

verb:[sem=[Tag=[action, grasp]], tag=Tag] --> grasp.
noun:[sem=[Tag=[thing, block]], tag=Tag] --> block.
noun:[sem=[Tag=[loc, table]], tag=Tag] --> table.
adj:[sem=[Tag=[colour, red]], tag=Tag] --> red.
spec:[] --> the.
prep:[sem=on] --> on.
```

Figure 4: Version of grammar from Figure 2 after transformation to AFF semantics.

representations, which is conditional mapping of lists to lists. Since AFF is an extension of bag-of-concepts, and bag-of-concepts is usually sufficiently unambiguous as it stands, we only want to add the functional markings in the cases where they are required. Most of our rules will thus still be `transfer_rule`s like the ones shown in Section 2, except that they now map lists of function-marking-tagged items to lists of function-marking-tagged items; however, in accordance with the stated design principles, we allow tags to be omitted when desired, with the convention that an omitted tag denotes an uninstantiated tag value.

One of the underlying linguistic intuitions behind AFF is that there are correspondences between functional markings in different languages, with each given functional marking f_s in the source language typically mapping to a specific functional marking f_t in the target language. For this reason, it would be highly unnatural only to specify transformations of tag values using `transfer_rule`s. We consequently introduce a second kind of rule, which we call a `tag_transfer_rule`; as the name suggests, this defines a direct mapping from tags to tags. Given the fact that functional tags have some claim to universality, it is reasonable to hope that many tags will map onto themselves. Thus a typical tag rule might map the English `subj` tag to the Arabic `subj` tag, which we write as

```
tag_transfer_rule(subj, subj).
```

Most tag transfer rules will be of the above simple form. However, there are always cases where languages diverge structurally, and here it will be necessary to make the tag transfer rule conditional on its surrounding context. For example, English constructions with the verb "last" ("Does the headache last more than ten minutes?") are realised differently in Arabic, using the transitive verb *tahus bi* ("feel"), thus here *hal tahus bi al soudaa li akthar min achr daqayq?* ("Do (you) feel the headache during more than ten minutes?"). Here, "headache" is marked as `subj` in English, but the correspoding Arabic word, *soudaa*, is the `obj` of *tahus bi*. We express the general fact that we wish to map `subj` to `obj` in the context of the verb "last" using the rule

```
tag_transfer_rule(subj, obj) :-
   context([state, last]).
```

We also require a normal `transfer_rule` which maps "last" to *tahus bi*. This also has to introduce an implicit second person subject, so the full rule is

```
transfer_rule(
  [[state, last]],
  [[state, tahus_bi],
   subj=[pronoun, anta]]).
```

(*anta* = "you"). Related sets of rules of this kind can be written more concisely with a small exten-

sion to the formalism, as follows:

```
transfer_rule(
  [[state, last]],
  [[state, tahus_bi],
   subj=[pronoun, anta]],
  [subj:obj]).
```

An important question we have so far postponed discussing is how to fill in unspecified tag values on the RHS of a `transfer_rule` application. At first, we believed that several possible strategies were feasible; rather to our surprise, examination of some examples convinced us that only one of these strategies actually made sense. The algorithm is as follows. We assume a `transfer_rule` R, whose LHS has successfully matched a set of tag/concept pairs, and consider the following cases:

1. R explicitly assigns values to all of the tags on its RHS. There is nothing more to do.

2. Not all of the tags on the RHS are assigned values by R. Apply `tag_transfer_rules` to all the matched tags on the LHS which were not originally assigned values by R, giving a set of tags $\{T_1...T_n\}$. There are now two subcases:

   (a) $n = 1$, i.e. only one transferred tag is produced. Set the values of all the uninstantiated tags on the transferred RHS to T_1.

   (b) $n > 1$, i.e. several different transferred tags are produced. Leave the values of the ininstantiated tags on the transferred RHS uninstantiated.

The least obvious part of this is (2a), which is easier to understand when we consider some more specific cases. The simplest and most common example is the case where R is a "lexical" `transfer_rule` which contains exactly one tag/concept pair on each side, each tag being left unspecified. We evidently need to apply a `tag_transfer` rule to the tag matched by the single pair on the LHS, to get the value of the tag attached to the transferred RHS.

To take a slightly more complex case, consider an English $\rightarrow$ Japanese rule which maps the expression "back of the head" to the single word *koutoubu*. We could write this as

```
transfer_rule(
  [[part, back],
   of=[body_part,head]],
  [[body_part, koutoubu]])
```

Here, it is clear that we want to translate the tag on the source-language pair that matches `[part, back]`, and assign it to the target-language element `[body_part, koutoubu]`. The translation of the tag `of` is irrelevant.

4 Using AFF in MedSLT

We have implemented and tested a version of AFF inside the Open Source MedSLT system, building AFF versions of the grammars for English, French Japanese, Arabic and the Interlingua. We also created AFF versions of the translation rules between the four surface languages and the Interlingua, in both directions. Coverage and performance of the two versions of the system on development data were essentially the same; the key differences were architectural in nature. We now briefly summarise these differences.

The basic tradeoff is between analysis and generation on one hand, and translation on the other. The more expressive AFF formalism implies that representations are less ambiguous, which means fewer problems in the analysis and generation components. The downside is that the translation rules become more complex. On the positive side, switching from bag-of-concepts to AFF allowed us to implement clean solutions to a substantial number of problems which were previously handled in an *ad hoc* manner. As previously noted, English constructions using verbs like "precede", "cause", "accompany", "visit" and "be in contact with" are in general ambiguous in the bag-of-words representation, and had to be solved by artificially constraining their arguments; AFF makes it possible to do this by simply differentiating between `subj` and `obj` tags. Similar considerations applied to constructions in the other two languages. For example, using bag-of-words, the Arabic frequency expressions *thalath marrat fi al ousbou* ("three times a week") and *marra kul thalathat assabii* ("once every three weeks") were previously represented in the same way, necessitating addition of a brittle generation preference. AFF once again allows the two expressions to be cleanly distinguished.

It was evident from the start that we would win on this kind of example; what was less clear

was the price we would have to pay, in terms of increased complexity of the transfer rule set. Gratifyingly, the conservative nature of the extension meant that this price turned out to be quite low. We had originally wondered whether it would be necessary to write many conditional `tag_transfer_rule`s, or add functional tags to a large proportion of the `transfer_rule`s. In fact, out of the total of 4444 rules used by the eight language pairs together, only 39 (0.9%) were conditional `tag_transfer_rule`s, and 524 (11.8%) were `transfer_rule`s containing at least one functional tag. A further 120 rules (2.7%) were unconditional `tag_transfer_rule`s. The remaining 3761 rules (84.6%) were `transfer_rule`s which did not explicitly mention functional tags, and were thus essentially bag-of-concepts mapping rules. To summarise, less than a sixth of the rules were affected by moving to the new framework.

5 Summary and Conclusions

We have described Almost Flat Functional semantics, a formalism which adds functional markings to a flat atheoretical feature/value representation. The additional functional information in AFF is sufficient to resolve nearly all of the representational ambiguities which caused problems for the flat bag-of-concepts formalism. In terms of representational complexity, however, the AFF formalism appears to be only slightly less tractable than bag-of-concepts. It seems reasonable to us that, like bag-of-concepts, it could also support learnable surface-oriented parsing; this could be combined with statistical recognition to provide a robust back-up to grammar-based speech processing (Rayner et al., 2005), a claim that we hope to investigate empirically in the near future. It is much less clear that full logic-based representations could be used for such purposes.

What we find interesting here, from a general perspective, is that we were able to create a reduced, but still essentially clean, form of a mainstream linguistic treatment, and incorporate it into an *ad hoc* engineering framework in a way that only marginally affected that framework's performance characteristics. Without wishing to exaggerate the importance of our results, we think examples like AFF suggest that the gulf between these two types of approach is not, perhaps, as wide as is sometimes suggested.

References

Bouillon, P., M. Rayner, N. Chatzichrisafis, B.A. Hockey, M. Santaholma, M. Starlander, Y. Nakao, K. Kanzaki, and H. Isahara. 2005. A generic multilingual open source platform for limited-domain medical speech translation. In *Proceedings of the 10th Conference of the European Association for Machine Translation (EAMT)*, pages 50–58, Budapest, Hungary.

Chatzichrisafis, N., P. Bouillon, M. Rayner, M. Santaholma, M. Starlander, and B.A. Hockey. 2006. Evaluating task performance for a unidirectional controlled language medical speech translation system. In *Proceedings of the HLT-NAACL International Workshop on Medical Speech Translation*, pages 9–16, New York.

Rayner, M., D. Carter, P. Bouillon, V. Digalakis, and M. Wirén, editors. 2000. *The Spoken Language Translator*. Cambridge University Press.

Rayner, M., P. Bouillon, N. Chatzichrisafis, B.A. Hockey, M. Santaholma, M. Starlander, H. Isahara, K. Kanzaki, and Y. Nakao. 2005. A methodology for comparing grammar-based and robust approaches to speech understanding. In *Proceedings of the 9th International Conference on Spoken Language Processing (ICSLP)*, pages 1103–1107, Lisboa, Portugal.

Rayner, M., B.A. Hockey, and P. Bouillon. 2006. *Putting Linguistics into Speech Recognition: The Regulus Grammar Compiler*. CSLI Press, Chicago.

Riezler, S., T.H. King, R.M. Kaplan, R. Crouch, J.T. Maxwell, and M. Johnson. 2002. Parsing the wall street journal using a lexical-functional grammar and discriminative estimation techniques. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (demo track)*, Philadelphia, PA.

Shieber, S., G. van Noord, F.C.N. Pereira, and R.C. Moore. 1990. Semantic-head-driven generation. *Computational Linguistics*, 16(1).

Wahlster, W., editor. 2000. *Verbmobil: Foundations of Speech-to-Speech Translation*. Springer.

Young, S. 2002. Talking to machines (statistically speaking). In *Proceedings of the 7th International Conference on Spoken Language Processing (ICSLP)*, pages 9–16, Denver, CO.