Syntactic and Semantic Input to Prosodic Markup in CommandTalk*

Elizabeth Owen Bratt
Center for the Study of Language and Information
Stanford University
Stanford, CA
ebratt@csli.stanford.edu

John Dowding
RIACS
NASA Ames Research Center
Moffett Field, CA 94035
jdowding@riacs.edu

Abstract

The CommandTalk military spoken dialogue system uses Semantic Head-Driven Generation in Gemini grammars, which have compositional semantics, permitting the identification of parse tree nodes corresponding to logical form subexpressions. This method of generation gives the CommandTalk prosody agent access to appropriate syntactic and semantic information to provide markup on generated sentences so that specialized military terms are pronounced correctly and so that general English language constructions important for dialogue can be given appropriate prosody to enhance their intelligibility and effectiveness. Specific military examples are map grid coordinates and unit call signs consisting of strings of letters and digits. The most salient English language construction supported is alternative questions, with contrastive stress on the distinguishing parts of alternatives.

Introduction

This paper describes how syntactic and semantic information is used in CommandTalk to improve intelligibility of the system's synthesized utterances. CommandTalk uses Semantic Head-Driven Generation to convert input logical forms into syntactic parse trees and surface strings. Both the parse tree and logical form are input to a *prosody agent*, which produces an annotated form of the surface string in the STML markup language, which is in turn input to the Festival speech synthesizer. We argue that an advantage of generation from logical forms, as opposed to statistical or template-based generation, is that it makes the syntactic parse tree available, and that prosodic mark-up based on the surface string combined with syntactic and semantic information is preferable to prosodic mark-up based on the surface string alone or simply syntactic or semantic information.

*This research was supported by the Defense Advanced Research Projects Agency under Contract N66001-94-C-6046 with the Space and Naval Warfare Systems Center, and performed while the authors were at SRI International. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official politicies, either express or implied, of the Defense Advanced Research Projects Agency of the U.S. Government.

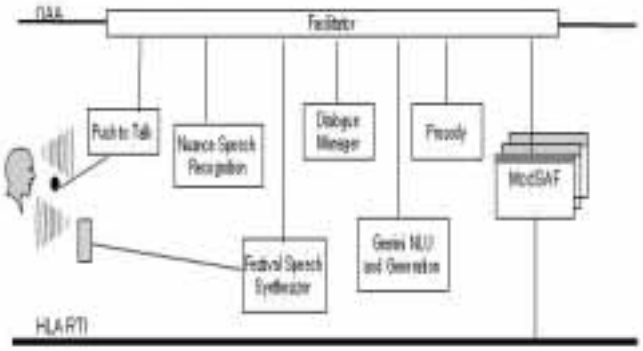

Figure 1: CommandTalk Architecture

Background

CommandTalk (Moore et al 1997, Stent et al. 1999) is a spoken dialogue interface to a battlefield simulator that allows operators to plan, layout, simulate, and observe military engagements. CommandTalk was first applied as a part of the Leathernet training system at the Marine Corps Air Ground Combat Center, and extended to support large scale simulation for all services in DARPA's STOW 97 exercise. Development of the dialogue portion of CommandTalk continued till the project's completion in 1999. Through composable finite state machines, the dialogue system supports clarification, confirmation, correction, reference resolution, and missing argument specification dialogues, which can be nested in subdialogues initiated by either system or user.

Figure 1 outlines the high-level architecture of CommandTalk, which consists of a number of independent components, combined using SRI's Open Agent Architecture (OAA) (Martin et al. 1998) . The OAA supports construction of complex software systems by allowing rapid combination of components, written in different programming languages, or for different operating system or hardware platforms. This encourages the use of existing commercial or publically available components. As each component is initialized, it registers its capabilities with a centralized *facili-*

tator, that manages the communication between the various components. The principal components are:

- Nuance - a commercially available continuous speaker-independent speech recognizer developed by Nuance Communications (Nuance 2002)
- Festival - an open-source speech synthesizer developed at the University of Edinburgh (Festival 2002)
- Gemini - a toolkit for spoken language understanding (Dowding et al. 1993) developed at SRI International. Gemini provides capabilities for parsing and interpreting the user's utterances, and generating the text for the system responses. Gemini grammars are written in a typed unification grammar formalism. Gemini includes capabilities to compile those grammars into Context-Free Grammars that can be used as language models by Nuance to constrain speech recognition (Moore 1998, Dowding et al. 2001, Rayner, Dowding and Hockey 2001)
- Dialogue Manager - maintains the representation of linguistic context, performs reference resolution, manages the interaction with the underlying battlefield simulation, and determines how to respond to user commands. The dialogue manager generates high-level logical form representations of the system's responses, which are passed to an *utterance planner* that provides more detailed information (such as number agreement, definite/indefinite determiner choice, referring expression generation) before being passed to Gemini for surface generation.
- Prosody - takes as input the logical form and the parse tree for the generated surface string, and produces an annotated STML (Spoken Text Markup Language, a precursor of the SABLE markup language) expression to be passed to Festival for synthesis.

Central to this architecture is the use of a single Gemini grammar to support parsing and interpretation of the user's utterances, generation of the system's responses, and, through a compiled form, as the language model for speech recognition. Gemini uses a variant of Semantic Head-Driven Generation (Shieber et al. 1990) . Of particular interest to this paper, Gemini uses compositional semantics, so that the logical form expression of a parent category in a grammar rule is constructed from the logical forms of its children in a way that does not depend on the particular values of those logical forms. In practical terms, this means that it is possible to traverse the logical form and the syntactic parse tree in parallel, and to reconstruct for each parse tree node the corresponding logical form subexpressions.

The architecture and many of the components of CommandTalk have been re-used in a number of more recent spoken dialogue systems, including the PSA Simulator (Rayner et al. 2000) , WITAS (Lemon et al. 2001) , a Naval damage control tutor (Clark et al. 2002), and SethiVoice (Gauthron and Colineau 1999).. The Gemini grammar written for CommandTalk was a semantic grammar, but some of these more recent systems have used more linguistically-motivated grammars.

Prosody Mark-up for Festival

CommandTalk customized its synthesized output to address particular characteristics of its often specialized military language. Because this specialized language did not occur in Festival training data, Festival did not give them an appropriately intelligible intonation. In addition to domain-specific intonation issues, there were also relatively domain-independent phenomena whose intonation could be improved. To accomplish these goals, a prosody agent used information from the syntactic parse tree in conjunction with the logical form interpretation of system utterances, and produced STML markup for the Festival speech synthesizer. Every system utterance had a syntactic parse tree and logical form interpretation available, because they were produced by generation from a Gemini grammar. We will describe three instances of intonation changes that were included in CommandTalk: appropriate pauses between military terms such as unit callsigns, list intonation in alternative questions, and contrastive stress.

Military Language

The main prosodic concern for system utterances in Commandtalk was to speak military terms in an appropriate way within a sentence. This required recognizing the military term as a constituent, plus respecting the internal structure of complex terms.

Grouping sequences of letters and digits The military language used in CommandTalk had several types of constituents which would be hard to understand if a large pause occurred within them, or a small pause occurred inside an internal constituent. Examples include map grid coordinates (*foxtrot quebec seven niner eight four one five*) and unit call signs (*one two zero alpha one one*). In each of these examples, the number of digits may vary, within the rules for constructing the terms. Without giving any futher guidance to Festival, these would not be grouped appropriately, and would often produce an unintelligible sequence of letters and digits.

Figure 1 shows a Gemini syntactic parse tree for a typical CommandTalk sentence. The terminals are in boldface. Each category is followed by a parenthesized, comma-separated list of its constituents. The relevant categories in this example are *grid_coordinates*, *sheet_id*, and *coordinate_nums*. Relying on the parse tree and these category names, the CommandTalk prosody agent adds STML markup to produce the system utterance: *M1 platoon proceed to foxtrot quebec <bound strength=2> seven nine eight <bound strength=3> four one five*. With these *bound* markings on either edge of the constituent of coordinates, Festival would likely insert any pauses at the marked bounds, rather than inside the constituent.

Because CommandTalk uses a semantic grammar, with category names such as *grid_coordinates*, the prosody agent can identify much of the needed information about special constructions needing prosodic markup purely by consulting the syntactic information in the parse tree. However, because the Gemini grammar allows the prosody agent to determine the logical form for each parse tree node, it would

sigma(utterance(nl_command(basic_nl_command(
unit(unit_nom(unit_n(echelon(echelon_id(echelon_type(
specific_echelon_type(tank_type(**m1**)),
echelon_noun(**platoon**))))))),
unit_command(basic_unit_command(movement_command(
movement_command(movement_verb(**proceed**)),
move_mod(objective_mod(to_prep(**to**),
engagement_loc(point_loc(coordinates(grid_coordinates(
sheet_id(any_letter(ica_letter(**foxtrot**)),
any_letter(ica_letter(**quebec**))),
coordinate_nums(coordinate_digits(digit(**seven**),
digit(**nine**), digit(**eight**)), coordinate_digits(digit(**four**),
digit(**one**), digit(**five**))))))))))))))))

Figure 2: Gemini parse tree for *M1 platoon proceed to foxtrot quebec seven nine eight four one five.*

be possible for the prosody agent to determine the need for markup based on the semantic information in the logical form, and then associate the semantic information with the expressed constituents in the parse tree.

English Language Issues

The CommandTalk prosody agent also added markup for general English language constructions, with alternative questions the most prominent example.

Alternative Questions Because the CommandTalk system asked alternative questions to resolve problems of ambiguity or underspecification in user commands, drawing the user's attention to the specific alternatives was important. An example question might be *There are three M1 tanks. Do you mean 152C11, 23B421, or 452W51?* Aiming to heighten the system's effectiveness, the prosody agent added markup for two characteristics of alternative questions: list intonation and contrastive stress.

List Intonation Alternative questions involve a list of alternatives, bearing ordinary list intonation. The prosody agent looks in the syntactic parse tree of an utterance for any list of daughters with a penultimate daughter realized by the word *or*, and adds markup for rising intonation, specifically *?*, at the end of each alternative daughter before the *or*.[1] The final alternative has the normal fall of a *wh* question or declarative statement.

Note that this markup would still be necessary regardless of the quality of the synthesis, since the surface strings of alternative questions are ambiguous with those of yes-no questions, while the intonations differ. For example, *Do you mean left or right?* is an alternative question if *left* and *right* are all the possible answers, but it is a yes-no question if the question is about whether the appropriate dimension is left-right or up-down. Thus looking to the logical form to determine the system intent is the best method of ensuring appropriate intonation markup. Generating from the Gemini grammar allows the prosody agent access to the semantic information of the logical form in conjunction with the syntactic information of the parse tree, so that all relevant dialogue information is available to contribute to prosodic markup.

Contrastive Stress Choosing an alternative from a list requires careful attention to the details of that list, which may not be necessary in statements about a list. Thus, the CommandTalk prosody agent consulted the logical form for a sentence to determine whether it was an alternative question. Alternative questions were indicated in the CommandTalk grammar by an *altq* wrapper around the rest of the logical form contents.

Within alternative questions, the prosody agent noted the differences in the word strings for each alternative within a list, as determined above, and put markup for emphasis around the contrasting parts. This helped draw attention to minor differences in longer strings, such as in the system utterance *Do you mean 100A***1***1 or 100A***2***1?*, which becomes the marked up *Do you mean one zero zero alpha <emph> one </emph> one?<bound strength=2> or one zero zero alpha <emph> two </emph> one*

This kind of detailed examination of syntactic differences between alternatives is something that straightforward template generation would not handle, since it requires a combination of knowing the overall construction plus the particular details of individual constituents within it. Thus, the placement of contrastive stress provides evidence that semantic information is not sufficient for placing syntactic markup, just as the falling final intonation of alternative questions provides evidence that syntactic information is similarly insufficient.

Related Work

The SOLE (Speech Output Labelling Explorer) concept-to-speech system (Hitzeman *et al.* 1999) uses also linguistic information from natural language generation to annotate text with XML tags for improving the prosody of synthesized system speech. The SOLE project values synthesizer-independent mark-up, as did our CommandTalk work. SOLE, however, aimed to annotate a wide range of linguistic constructs as a basis for statistical methods on an annotated corpus to determine the appropriate mapping to intonation, while CommandTalk involved rule-based methods to support clarity in typical utterances in our dialogues.

In a discussion of how natural language generation can provide information to compute prosody in a concept-to-speech system, Theune (2002) focusses on the automatic marking of contrastive accents. Her work used the D2S system for spoken language generation, which takes data as input, and produces enriched text, annotated with prosodic markers for accents and phrase boundaries, and next, a speech signal. Both syntactic and semantic information are used in the Prosody module in D2S.

McKeown and Pan (1999) explore methodology for automatic learning of the correlation between linguistic features and prosody in concept-to-speech systems, such as

[1] We are grateful to Alan Black for providing us with a patch to Festival to enable this rising intonation and its association with the *?* symbol.

their MAGIC system for multimedia briefings of the status of bypass patients.

A distinguishing characteristic of the CommandTalk work described here is that the natural language generation uses the same grammar as the parsing and interpretation component. Using this single grammar, the CommandTalk prosody agent has access to sufficient syntactic and semantic detail to provide the necessary prosodic mark-up. Taking this approach lays the groundwork for a future system which would interpret recognized speech including prosodic detail, making the correspondence between linguistic constructs and prosodic detail bidirectional, in the same way as the bidirectional correspondence of logical forms and word strings.

Conclusion

In this paper we describe the prosody agent in CommandTalk, and how it used syntactic and semantic information to add appropriate intonational markup, and to improve the intelligibility of the synthesized responses. Grammar-guided generation allows the prosody agent access to the appropriate kind of information for handling various phenomena accurately and precisely.

References

Clark, B.; Bratt, E. O.; Lemon, O.; Peters, S.; Pon-Barry, H.; Thomsen-Gray, Z.; and Treeratpituk, P. 2002. A general purpose architecture for intelligent tutoring systems. In *the Proceedings of the International CLASS Workshop on Natural, Intelligent, and Effective Interaction in Multimodal Dialogue Systems*.

Dowding, J.; Gawron, M.; Appelt, D.; Cherny, L.; Moore, R.; and Moran, D. 1993. Gemini: A natural language system for spoken language understanding. In *Proceedings of the Thirty-First Annual Meeting of the Association for Computational Linguistics*.

Dowding, J.; Hockey, B. A.; Culy, C.; and Gawron, J. M. 2001. Practical issues in compiling typed unification grammars for speech recognition. In *Proceedings of the Thirty-Ninth Annual Meeting of the Association for Computational Linguistics*.

Festival. 2002. *The Festival Speech Synthesis Systems*. http://www.cstr.ed.ac.uk/projects/festival. As of 14 October 2002.

Gauthron, O., and Colineau, N. 1999. SETHIVoice: Cgf control by speech-recognition/interpretation. In *I/ITSEC '99 (Interservice/Industry Training, Simulation and Education Conference), Synthetic Solutions for the 21st Century*.

Hitzeman, J.; Black, A. W.; Taylor, P.; Mellish, C.; and Oberlander, J. 1999. An annotation scheme for concept-to-speech synthesis. In *Proceedings of the European Workshop on Natural Language Generation*, 59–66.

Lemon, O.; Bracy, A.; Gruenstein, A.; and Peters, S. 2001. A multi-modal dialogue system for human-robot conversation. In *Proceedings of North American Association for Computational Linguistics (NAACL 2001)*.

Martin, D.; Cheyer, A.; and Moran, D. 1998. Building distributed software systems with the open agent architecture. In *Proceedings of the Third International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology*.

McKeown, K., and Pan, S. 1999. Prosody modeling in concept-to-speech generation: Methodological issues. *Philosophical Transactions of the Royal Society, Series A*.

Moore, R.; Dowding, J.; Bratt, H.; Gawron, J.; Gorfu, Y.; and Cheyer, A. 1997. CommandTalk: A spoken-language interface for battlefield simulations. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, 1–7.

Moore, R. 1998. Using natural language knowledge sources in speech recognition. In *Proceedings of the NATO Advanced Studies Institute*.

Nuance. 2002. http://www.nuance.com. As of 14 October 2002.

Rayner, M.; Dowding, J.; and Hockey, B. 2001. A baseline method for compiling typed unification grammars into context free language models. In *Proceedings of Eurospeech 2001*, 729–732.

Rayner, M.; Hockey, B.; and James, F. 2000. A compact architecture for dialogue management based on scripts and meta-outputs. In *Proceedings of Applied Natural Language Processing (ANLP)*.

Shieber, S.; van Noord, G.; Moore, R.; and Pereira, F. 1990. A semantic head-driven generation algorithm for unification grammars. *Computational Linguistics* 16(1).

Stent, A.; Dowding, J.; Gawron, J.; Bratt, E. O.; and Moore, R. 1999. The CommandTalk spoken dialogue system. In *Proceedings of the Thirty-Seventh Annual Meeting of the Association for Computational Linguistics*, 183–190.

Theune, M. 2002. Contrast in concept-to-speech generation. *Computer Speech and Language* 16:491–531.