Interpreting Language in Context in CommandTalk

John Dowding, Elizabeth Owen Bratt and Sharon Goldwater
SRI International
{dowding,owen,goldwater}@ai.sri.com

February 5, 1999

1 Introduction

CommandTalk [6] is a spoken language interface to the ModSAF [1] battlefield simulator that allows simulation operators to generate and execute military exercises by creating forces and control measures, assigning missions to forces, and controlling the display. ModSAF is a stochastic distributed entity-based simulation, where one operator controls a collection of forces, while other operators may simultaneously control other allied and/or opposing forces. This paper will focus on how two representations of context are used in CommandTalk to correctly interpret the user's spoken utterances: *situational context* represents the current state of the simulation, and *linguistic context* represents the history of the user's linguistic acts.

2 Background

CommandTalk consists of a number of independent, cooperating *agents* interacting through SRI's Open Agent Architecture (OAA) [4]. This architecture allows separate components to be developed independently, then flexibly and dynamically combined to support distributed computation. The principal components of CommandTalk are:

- Speech Recognition - based on Nuance, a commercial continuous-speech speaker-independent speech recognizer developed from SRI technology. Its language model is compiled [5] from the Gemini grammar used by the Natural Language agent.
- Natural Language - based on Gemini [2], performs syntactic and semantic analysis [3] and text generation [7].
- Contextual Interpretation (CI) - This agent is described in detail in section 3.
- Speech Synthesis - based on the Festival speech synthesizer[1] developed by the Centre for Speech Technology Research (CSTR) at the University of Edinburgh.
- Prosody - annotates system utterances with cues in the Spoken Text Markup Language[2] (STML) to inform the speech synthesizer about pauses and pitch changes.

[1]See `http://www.cstr.ed.ac.uk/projects/festival.html` for full information on Festival.
[2]See `http://www.cstr.ed.ac.uk/projects/ssml.html` for details.

- ModSAF - accepts messages from the CI agent, and translates them into queries and commands in the ModSAF Agent Layer Language[3].

3 Language in Context

This section will discuss some ways in which the CI agent uses context to properly interpret the user's commands. CommandTalk relies on the ModSAF agent to provide it with a representation of events in the simulated world. The CI agent uses this *situational context* to, among other things, resolve references, interpret commands with respect to a unit's mission, and satisfy parameters of underspecified commands. Similarly, the user's utterances can not be taken in isolation, but needs to be interpreted within the *linguistic context* of an ongoing dialogue with the system.

3.1 Situational Context

The ModSAF agent sends messages to the CI agent whenever a new simulation object is created, modified or destroyed. The message stream contains simulation objects that correspond to "physical" objects (e.g. units, shells, and radio messages) and information about the physical attributes of the object (e.g. type, size, location, and appearance). For objects that are under the user's command, further information is available, including control measures (points, lines, or areas of strategic significance), command hierarchy (the structure of companies, platoons, and squads), and the status of current and planned missions. Simulation objects that are not under the user's command can still occur within spoken commands, such as when they are being attacked, pursued, or resupplied.

Reference resolution in CommandTalk determines what objects and sets of objects in the simulated world correspond to the user's linguistic descriptions. The range of references supported in CommandTalk include singular noun phrases, such as proper names (*A11*), definite descriptions (*M1 tank platoon*), and pronouns (*it*); as well as plural noun phrases, such as plural descriptions (*M1 tank platoons*), quantified descriptions (*all friendly forces*), conjunctions (*A11 and B11*), and pronouns (*them*). To determine the correct reference, the CI may also need to take into account optional gestural information, such as *these units* when accompanied by a pointing or circling gesture, or salience, such as *that unit* when a unit has just appeared on the user's console.

Commands given to forces under the user's command need to be interpreted in the context of the current and planned missions for the given unit. For instance, a simple command like *A11 advance to Checkpoint 1* can be given a variety of meanings depending on the context:

- If A11 is not currently executing a mission, add a task to move to Checkpoint 1 as a continuation of the A11's existing plan, waiting to commence moving there until a *move out* command is given.
- If All has a pending task to move to Checkpoint 1, carry out that task (equivalent to a *move out* command).
- If All is currently executing a mission, temporarily override that mission by moving to Checkpoint 1 right away, leaving the underlying mission to be resumed later.

[3]See `http://www.ai.sri.com/~lesaf/mall.html`.

- Conversely, if A11 has suspended a mission involving moving to Checkpoint 1, resume executing that underlying mission.
- If A11 is currently executing a mission, completely replace that mission, if the mission is not central to the commander's intent.

Users may express any given command in a variety of ways. Consider the following examples of the **attack by fire** command:

U 1 Establish a base of fire at Checkpoint 2 facing Objective Alpha.
U 2 Move to Checkpoint 2 and attack the enemy with direct fire.
U 3 Engage the enemy to the north
U 4 Action right

An **attack by fire** command has two required arguments: a position to attack from, the **base of fire**; and a position to fire at, the **engagement area**. But, these examples show that the user's language may not always express those arguments directly. While (1) contains both arguments, (2) requires the system to identify Checkpoint 2 as the base of fire, and presupposes that *the enemy* can be resolved from context. In (3) and (4), the base of fire is omitted completely, and must be determined from the planned location of the unit when the command is executing, and the engagement area is specified only vaguely. This introduces a problem similar to reference resolution, where the values of arguments may be omitted or underspecified, and the system needs to induce those values from context.

3.2 Linguistic Context

The CI agent carries out three subtasks of dialogue management: managing the representation of linguistic context, interpreting the user's utterances within that context, and planning and generating the system's spoken responses (see [8]). This section will concentrate on issues of interpretation within linguistic context.

Reference resolution normally determines the relationship between linguistic descriptions and objects in the simulated world, but references can also refer to linguistic descriptions themselves:

U 5 A11 advance to Objective B.
S 6 There is no Objective B. Which point should 100A11 proceed to?
U 7 Create it at 635 545.
S 8 Should 100A11 proceed to Objective B?
U 9 Yes.

In this example, the system must realize that *it* refers to the description of Objective B, not Objective B itself. Notice also that there is an interplay between situational context and linguistic context. In (8) the system realizes that the simulated world has changed, and that the command expressed in (5) can now be executed.

The system's representation of linguistic context is sophisticated enough to recognize nested dialogue structure, and allow a user to engage in a subdialogue before answering a question.[4]

[4] A user who answers a question from the system directly can reply with just a noun phrase. If the user chooses to engage in other dialogue before answering, then the reply must contain enough information to link it with the system's earlier question.

U	11	A13 proceed to Checkpoint 1 in a column formation.
S	12	There is no A13. Which unit should proceed in a column formation to Checkpoint 1?
U	13	What units are there?
S	14	There are 2 units: 100A11 and 100A12.
U	15	A11 proceed to Checkpoint 1.
S	16	100A11 will proceed in a column formation to Checkpoint 1.

In this case, although (15) is completely interpretable on its own, the system correctly identifies it as the answer to the question posed in (12).

References

[1] Andrew Ceranowicz. Modular Semi-Automated Forces. In J.D. Tew et al., editor, *Proceedings of the Winter Simulation Conference*, pages 755–761, 1994.

[2] John Dowding, Jean Mark Gawron, Doug Appelt, Lynn Cherny, Robert Moore, and Douglas Moran. Gemini: A Natural Language System for Spoken Language Understanding. In *Proceedings of the Thirty-First Annual Meeting of the ACL*, Columbus, OH, 1993. Association for Computational Linguistics.

[3] John Dowding, Robert Moore, Francois Andry, and Douglas Moran. Interleaving Syntax and Semantics in an Efficient Bottom-Up Parser. In *Proceedings of the Thirty-Second Annual Meeting of the ACL*, Las Cruces, New Mexico, 1994. Association for Computational Linguistics.

[4] David Martin, Adam Cheyer, and Douglas Moran. The Open Agent Architecture: A framework for building distributed software systems. *Applied Artificial Intelligence*, 13(1–2), January–March 1999.

[5] Robert Moore. Using Natural Language Knowledge Sources in Speech Recognition. In Keith Ponting, editor, *Speech Pattern Processing*. Springer-Verlag, 1999.

[6] Robert Moore, John Dowding, Harry Bratt, J. Mark Gawron, Yonael Gorfu, and Adam Cheyer. Commandtalk: A spoken-language interface for battlefield simulations. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, pages 1–7, Washington, DC, 1997. Association for Computational Linguistics.

[7] S. M. Shieber, G. van Noord, R. Moore, and F. Pereira. A semantic head-driven generation algorithm for unification-based formalisms. *Computational Linguistics*, 16(1), March 1990.

[8] Amanda Stent, John Dowding, Jean Mark Gawron, Elizabeth Owen Bratt, and Robert Moore. The CommandTalk Spoken Dialogue System. In *Submitted to the Thirty-Seventh Annual Meeting of the Association for Computational Linguistics*, College Park, Maryland, 1999.