Towards Hands-Free Human-Robot Interaction through Spoken Dialog

Vladimir A. Kulyukin
Computer Science Department
Utah State University
Logan, UT 84322-4205
vladimir.kulyukin@usu.edu

Abstract

We present an approach to hands-free interaction with autonomous robots through spoken dialog. Our approach is based on passive knowledge rarefication through goal disambiguation, a technique that allows the robot to refine and acquire knowledge through spoken dialog with a human operator. A key assumption underlying our approach is that the operator and the robot share the same set of goals. Another key idea is that language and vision have some memory structures in common. We discuss how our approach achieves four types of human-robot interaction: command, elaboration, introspection, and instruction-based learning. We discuss our experiences with implementing our approach on an autonomous robot.

Introduction

Research on human-robot interaction has come to the forefront of autonomous robotics. The main motivation for this line of inquiry is the need for autonomous robots that collaborate with people in complex environments on a variety of tasks ranging from mining to navigation to space walks (Fong and Thorpe 2001).

Most current approaches to human-robot interaction are based on graphical user interfaces (GUI's)(Cheng and Zelinsky 2001). The robot's abilities are expressed in a GUI through graphical components. A human operator sends a message to the robot through a GUI event, i.e., a menu selection. The operator's message initiates an action by the robot. The robot sends feedback to the GUI, and the interaction cycle repeats.

GUI-based approaches are appropriate in situations where the operator has access to a monitor or a keyboard, e.g., an engineer monitoring the work of a mining robot or an astronaut monitoring the work of a space shuttle arm. In many situations, however, access to monitors or keyboards is not available. An injured person in an urban disaster area is unlikely to communicate with a rescue robot through a GUI. A geologist exploring the surface of Mars will benefit from a robotic assistant that can interact with the geologist without having the latter touch a monitor or type on a keyboard.

When the GUI-based approach is not feasible, speech is one obvious alternative. The principal challenge here is the integration of speech with the robot's other abilities, especially vision. For, if speech is to control every aspect of the robot's behavior, the cognitive load on the operator becomes unbearably high.

In this paper, we present an approach to hands-free interaction with an autonomous robot through spoken dialog. Our approach is based on passive knowledge rarefication through goal disambiguation, a technique that allows the robot to refine and acquire knowledge through spoken dialog with a human operator. A key assumption underlying our approach is that the operator and the robot share the same set of goals. This assumption is justified on empirical grounds: communication is of little value if the operator's goals have nothing in common with the robot's. Another key idea behind our approach is that language and vision share some common memory structures. Since most of everyday language use is grounded in concrete, visually-guided activity, language and vision must have at least some memory structures in common. Otherwise, it would be hard, if not impossible, for us to understand and follow instructions or to relate spoken utterances to external objects.

The remainder of our paper is organized as follows. Section 2 describes the robot's hardware and architecture. Section 3 concerns the robot's knowledge representation. Section 4 presents four types of hands-free human-robot interaction and explains how the robot engages in them. Section 5 discusses related work, the lessons learned, and current limitations of our implementation. Section 6 offers our conclusions.

Hardware and Architecture

Our autonomous robot is a Pioneer 2DX robot assembled from the robotic toolkit from ActivMedia, Inc. (www.activmedia.com) (see Figure 1).The robot patrols an office area looking for pieces of trash, such as soda cans, coffee cups, and crumpled pieces of paper, picking them

Figure 1: Pioneer 2DX Robot.

Figure 2: Robot Avoiding a Person.

up, and carrying them to designated areas on the floor. The robot also accepts voice inputs from human operators in different offices. For example, operators can inform the robot that there is a soda can in front of a specific office. As seen in Figure 2, the robot's environment is relatively complex inasmuch as the robot is not the only actor and has to deal with dynamic contingencies such as avoiding people and dealing with closed doors. The robot has three wheels, an x86 200 MHZ onboard computer with 32MB of RAM running Windows NT 4.0, a gripper with two degrees of motion freedom, and a pan-tilt-zoom camera. The camera has a horizontal angle of view of 48.8 degrees and a vertical angle of view of 37.6 degrees. The images are saved as 120 by 160 color bitmaps with the Winnov image capture card (www.winnov.com). The other hardware components include a CCTV-900 wireless AV receiver and switcher, an InfoWave radio modem, and a PSC-124000 automatic battery charger.

The robot features the three-tier (3T) architecture (Bonasso et al. 1997). The architecture consists of three tiers of functionality: deliberation, execution, and sensory-motor skills. The deliberation tier is responsible for symbolic processing, e.g., planning and language understanding; the execution tier manages task networks of behaviors; the sensory-motor tier interacts with the world. The 3T architecture was chosen because, in our opinion, it offers a more flexible interaction between different tiers of functionality than its hierarchical (Nilsson 1969) and reactive (Chapman 1991) counterparts. More specifically, the 3T architecture offers a reasonable compromise between deliberation and reactivity: it builds on reactive models by tightly coupling sensing to action, but considers deliberation as an action available to the robot on demand, e.g., when a robot cannot make sense of sensory inputs. The execution tier is implemented with the Reactive Action Package (RAP) system (Firby, Prokopowicz, and Swain 1995). A RAP is a set of methods for achieving goals under different circumstances. For example, a mobile

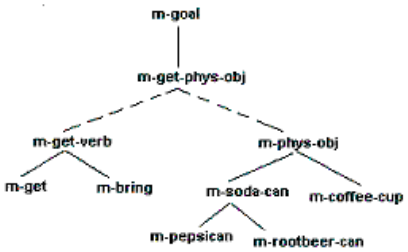

Figure 3: Semantic Network.

robot can have two methods in a RAP for homing in on an object. One method assumes that the robot knows where the object is. The other method makes no such assumption and instructs the robot to find it first. The RAP system consists of the RAP programming language and an interpreter for executing RAPs written in that language.

The skill system for the robot is described in detail elsewhere (Kulyukin and Morley 2002; Kulyukin and Steele 2002). Here we offer only a brief summary. The skill system consists of obstacle avoidance skills, navigational skills, and object recognition skills. Object recognition skills use template-based object matching routines based on color histogram matching and the Generalized Hamming Distance (Bookstein, Kulyukin, and Raita 2002). Skills are enabled and disabled by the RAP interpreter. The output of a skill is defined as a set of symbolic predicate templates that are instantiated with specific values extracted from sensory data. These predicate templates allow the robot to transform raw sensory data from the world into symbolic assertions about it. The skill system also interfaces to Saphira, a software library for controlling the Pioneer robot's hardware, e.g., reading sonars, setting rotational and translational velocities, etc.

Knowledge Representation

The deliberation tier of our 3T architecture is a semantic network of goals, objects, and actions. The semantic network is the robot's memory that shapes the robot's interaction with its environment. Figure 3 shows a small part of the semantic network. Each node in the network is a memory organization package (MOP) (Martin 1993) and, therefore, starts with the **m-** prefix. The solid lines correspond to the *abstraction* links; the dotted lines denote the *packaging* links. For example, the **m-get-phys-obj** node, which corresponds to the goal of getting a physical object, has two *packaging* links: the first one links to **m-get-verb** and the second one links to **m-phys-obj**. The two links assert that the MOP **m-get-phys-obj** has two typed slots: one of type **m-get-verb** and the other of type **m-phys-obj**. The semantic network links each concrete object node with its models used by object recognition skills. Thus, the robot's memory brings together symbolic and but also visual information.

Nodes in the network are activated by a spreading activation algorithm based on direct memory access

parsing (DMAP) (Martin 1993). The actual activation is done through *token sequences* obtained from speech recognition (Kulyukin and Steele 2002). A token is a symbol that must be directly seen in or activated by the input. Token sequences are associated with nodes in the semantic network. For example, **m-pepsican** has two token sequences associated with it **{a pepsican}** and **{the pepsican}.** Thus, if an input contains either sequence, **m-pepsican** is activated. Token sequences may include direct references to *packaging* links. For example, the only token sequence associated with **m-get-phys-obj** is **{get-verb-slot phys-obj-slot}**, which means that for **m-get-phys-obj** to be activated, **m-get-verb** must be activated first and then **m-phys-obj** must be activated.

To keep track of token sequences at run time, we use the concept of *expectation* (Kulyukin and Settle 2001). An expectation is a 3-tuple consisting of a node, a token sequence, and the next token in the token sequence that must be seen in the input. For example, **X = <m-pepsican, {a pepsican}, a>** is a valid expectation. The first element of an expectation is referred to as its *target*, while the third element is referred to as its *key*. Thus, **Target(X) = m-pepsican**, **TokenSeq(X) = {a pepsican}**, and **Key(X) = {a}**. If the next token in the input is **a**, **X**'s token sequence can be *advanced*, thus making **X** into **<m-pepsican, {pepsican}, pepsican>**. The token **pepsican** will cause **X** to become **<m-pepsican, {}, null>**. When an expectation's token sequence becomes empty, the expectation's target is *activated*.

The procedural knowledge is represented through RAPs. Consider the following RAP that specifies how the robot disposes of a trash item in its gripper.

```
(define-rap
(index (dump-trash ?dist ?speed))
(method
 (task-net
  (sequence
   (t1 (robot-deploy-gripper-prim))
   (t2 (robot-move-prim 0 ?speed  ?dist)
   (t3 (robot-back-up-prim ?speed ?dist)
   (t4 (robot-store-gripper-prim))
  ))))
```

The above RAP triggers the following behavior. The robot first lowers and opens the gripper. It then moves forward a given distance at a given speed, pushing the trash item across the floor in front of its gripper. After the move is complete, the robot backs up and stores the gripper in its original position.

The procedural knowledge is connected to the declarative knowledge through the mechanism of *callbacks* (Martin 1993; Kulyukin and Settle 2001). A callback is a procedure associated with a node in the semantic network. A callback runs when the node it is associated with is activated. For example, the semantic network has a node **m-dump-trash**. When **m-dump-trash** is activated, the callback associated with that node allows the robot to reference the **dump-trash** RAP.

Figure 4: Local Mode of Interaction.

Interaction

There are two modes of interaction between the operator and the robot: *local* and *remote*. The local mode is depicted in Figure 4. In the local mode, the robot and the operator are within visual proximity of each other. The operator wears a head microphone and communicates with the robot only through speech. The speech recognition engine we use is Microsoft's Speech API (SAPI) 5.1 (www.microsoft.com/speech). Since the only sound the robot can generate is a beep, the responses from the robot are synthesized into speech on the operator's computer. In Figure 4, the operator's speech engine runs on an offboard computer connected to the robot's computer through wireless Ethernet. If it is necessary for the operator to follow the robot, the speech recognition engine is mounted on a laptop that the operator carries around. In the remote mode, neither the operator nor the robot can see each other, so the operator relies only on the video feed from the robot's camera. The interaction between the operator and the robot is based on the passive rarefication of knowledge through goal disambiguation. The robot uses the operator's speech input to reference a goal in memory directly. If the input references a goal completely, the robot activates behaviors necessary to achieve it. If the input gives a partial description of a goal, the robot treats the goal as ambiguous and attempts to disambiguate it through dialog. If no goal is referenced, either completely or partially, the robot uses the input to rarefy an active ambiguous goal. For example, if the input specifies a color and an active goal is to get a soda can, the robot rarefies the goal to getting a red soda can. If the goal rarefication fails, the robot tells the operator that speech is not understood. Knowledge acquisition occurs when the operator tells the robot how to modify a behavior. We will say more on knowledge acquisition later.

The algorithm is passive in that the robot never initiates an interaction with the operator unless asked to achieve a goal. This distinguishes our approach from dialog-based mixed-initiative approaches where autonomous agents act proactively (Rich, Sidner, and Lesh 2001). It is our view that when the robot and the operator share common goals, the complex problems of dialog state tracking and intent recognition required in proactive approaches can be safely avoided. The four broad classes of interaction that the operator and the robot engage in through spoken dialog are command, elaboration, introspection, and instruction-based learning. Below we cover each class in greater detail.

Figure 5: An Image with a Pepsican.

Command

Command is the type of interaction that occurs when the operator's input references a goal in the robot's memory unambiguously. For example, when the operator asks the robot to pick up a pepsican in front of the robot, the robot can engage in the pickup behavior right away. Command is a minimum requirement for human-robot dialog-based interaction in that the robot must at least be able to execute direct commands from the operator.

Let us go through an example. In addition to showing how the robot executes commands, this example underlines an important principle of our architecture: the execution tier directly manipulates the robot's memory in the deliberation tier.

Consider an image in Figure 5. Suppose the operator's command to the robot at this point is "Get the pepsican." One of the token sequences associated with **m-get-phys-obj** is **{get-verb-slot phys-obj-slot}**. In other words, to activate **m-get-phys-obj**, an input must first activate the verb slot of the MOP and then the physical object slot. The **m-get** node's token sequence is **{get}**, thus the token **get**, the first token in the operator's command, activates **m-get**, which, in turn, activates **m-get-verb**. Since the verb slot of **m-get-phys-obj** is activated, the token sequence of **m-get-phys-obj**'s expectation is advanced one token to the right and becomes **{phys-obj-slot}**. The tokens obtained from "the pepsican" activate **m-pepsican** and, consequently, **m-phys-obj**, thus activating **phys-obj-slot**. As soon as **m-pepsican** is activated, the object models for pepsicans are brought into the robot's memory unless, of course, they have not been activated by prior dialog. After **m-phys-obj** is activated, the token sequence of **m-get-phys-obj**'s expectation becomes empty, thus activating **m-get-phys-obj**. When **m-get-phys-obj** is activated, an instance MOP, call it **m-get-phys-obj-12**, is created under it with values **m-get** for the verb slot and **m-pepsican** for the object slot. The new MOP is placed under the **m-active-goal** to reflect the fact that it is now one of the active goals pursued by the robot. In addition, when **m-get-phys-obj** is activated, its callback installs the following goal on the RAP interpreter's agenda: **(refine-and-execute m-get-phys-obj)**. The RAP for achieving this goal checks for instances of **m-get-phys-obj** that are also instances of **m-active-goal**. In this case, the RAP finds **m-get-phys-obj-12**, extracts the **m-pepsican** value from the frame's object slot and installs the following goal on the agenda: **(get-phys-obj m-pepsican)**. When executed, the **get-phys-obj** RAP enables **detect-obj-skill**. The skill captures the image shown in Figure 5, detects the pepsican, and puts the following assertions in the RAP memory: **(detected-obj pepsican)**, **(dist-to 1.5)**, **(obj-xy 81 43)**, and **(sim-score .73)**. These assertions state that the skill detected a pepsican 1.5 meters away from the robot, and the bottom left coordinates of the image region that had the best matching score of .73 are x=81, y=43. The active RAP, i.e., **get-phys-obj**, then invokes the homing skill. The skill calculates the angle by which the robot must turn from the bottom left coordinates and causes the robot to navigate to the object. When the robot is at the object, the RAP enables the pickup skill so that the robot can pick up the can with the gripper. Finally, the RAP removes **m-get-phys-obj-12** from under **m-active-goal**.

Elaboration

The robot engages in elaboration when a goal received from the operator is underspecified or ambiguous. There are two types of elaboration: *pre-visual* and *post-visual*. To understand how these types are different, consider a situation when the robot is looking at multiple objects. Suppose the operator asks the robot to get a soda can. As discussed above, the operator's speech will activate **m-get-phys-obj**. Since the robot knows several types of soda cans, the goal is underspecified in the sense that it does not tell the robot the *type* of soda can the operator wants it to get. Hence, the robot asks the operator to elaborate on the type of can. When the operator specifies the type of soda can, e.g. pepsican, the robot proceeds as in the previous example to detect the can, navigate to it, and pick it up. Note that the robot asks the operator to elaborate the goal before doing any visual processing. Hence, the term *pre-visual*. Pre-visual elaboration saves the robot some potentially expensive visual processing.

Post-visual elaboration occurs when a goal is ambiguous with respect to a set of symbolic assertions obtained from visual processing. Suppose that the operator asks the robot what objects it sees when the robot is looking at multiple objects. After doing the visual processing, the robot's memory contains assertions about a pepsican, a root beer can, and a crumpled piece of paper. Suppose that now the operator asks the robot to get a soda can. Since the robot has detected two cans, the goal is ambiguous and the robot asks the operator to elaborate on which can must be picked up.

Introspection and Learning

Introspection and learning are closely related. Introspection allows the operator to ask the robot about its knowledge. There are three types of introspective questions the robot can handle: questions about the robot's past or current state, questions about the robot's knowledge of objects, and questions about the robot's behaviors. Questions of the first type, e.g., "What are you doing?" or "Did/Do you see a pepsican?" are answered through direct examination of the semantic network or the RAP memory. For example, when asked what it is doing, the robot generates natural language descriptions of the active goals

currently in the semantic network. Natural language generation is done on the basis of descriptions supplied by the RAP programmer. We extended the RAP system specification to require that every primitive RAP be given a short description of what it does. Given a RAP, the language generation engine recursively looks up descriptions of each primitive RAP and generates the RAP's description from them. For example, when asked the question "How do you dump trash?", the robot generates the following description from the **dump-trash** RAP: "Deploy the gripper. Move forward a given distance at a given speed. Back up a given distance at a given speed. Store the gripper."

The only kind of learning the robot does is instruction-based. It occurs when the operator instructs the robot to modify a RAP on the basis of the generated description. The modification instructions include the removal of a task, insertion of a task, and combination of several tasks. For example, given the above description from the robot, the operator can issue the following instruction to the robot: "Back up and store the gripper in parallel." The instruction causes the robot to permanently modify the last two tasks in the **dump-trash** RAP to run in parallel:

```
(define-rap
(index (dump-trash ?dist ?speed))
(method
 (task-net
  (sequence
   (t1 (robot-deploy-gripper-prim))
   (t2 (robot-move-prim 0 ?speed ?dist))
   (parallel
    (t3 (robot-back-up-prim ?speed ?dist)
    (t4 (robot-store-gripper-prim))))
  ))))
```

Discussion

Natural language interfaces to autonomous robots have an extensive research record in AI. The possibility of a dialog with the robot was first suggested by Nilsson (1969) in his work on Shakey. Winograd's SHRUDLU (Winograd 1972) simulated a dialog with a virtual robot. Chapman's work on Sonja (Chapman 1991) investigated instruction use in reactive agents. Torrance (1994) investigated natural language communication with a mobile robot performing odometry-based navigational tasks. Horswill (1995) integrated a SHRUDLU-like instruction processing system with a real-time vision system. Fitzgerald and Firby (2000) proposed a RAP-based reactive task architecture for dialog systems.

Our approach builds on and complements these approaches in five ways. First, our approach is based on the assumption that the robot and the operator share the same set of goals. This makes it unnecessary to maintain dialog histories or do dynamic plan recognition and generation (Rich, Sidner, and Lesh 2001), which is expensive at best and intractable at worst. Chapman (1991) and Horswill (1995) make similar claims but their agents do not reason about goals. On the other hand, Chapman and Horswill present arguments for biological plausibility, whereas we do not. Second, our approach claims that language and vision, at least in its late stages, share some common memory structures. In effect, we state that late vision is part of cognition. Horswill's Ludwig also grounds language interpretation in vision but makes no claims about memory structures used by vision and language. Chapman makes a claim similar to ours but does not go beyond simulation in implementing it. Third, we present some computational machinery for introspection and instruction-based learning. Of the mentioned approaches only Torrance's (1994) and Fitzgerald and Firby's (2000) systems could, in our opinion, be extended to handle introspection and learning. It is difficult to see how the reactive architectures advocated by Chapman and Horswill can handle introspection and learning even in principle, since they lack internal states. Fourth, our approach is similar to Torrance's and Fitzgerald and Firby's in that it offers dialog capabilities that go beyond GUI's. Nilsson's Shakey (Nilsson 1969), Chapman's Sonja, and Horswill's Ludwig also use natural language inputs. However, natural language interactions in these systems are so simple that they could be easily replaced with mouse menus. Finally, we commit to spoken dialog, whereas all of the above approache use typewritten inputs, thus prohibiting hands-free interaction between the robot and the operator.

By implementing our approach on an autonomous robot we have learned several lessons for designing autonomous robots capable of hands-free interaction with humans through spoken dialog. The first lesson is that interaction between robots and humans goes beyond GUI's. Although our initial attempt was GUI-based, we soon realized that GUI's were putting too many artificial constraints on human-robot interaction. The second lesson is that the robot and the operator must share the same set of goals. In general, for any meaningful communication to take place, it must be assumed that the operator and the robot share some common reality. The third lesson is that the integration of language and vision through the same memory structures seems to be necessary to support content-rich dialog between the robot and the operator. This is because the robot must be able to carry a lot of perceptual work on its own. If that were not the case, the operator would have to tell the robot exactly what to do and how to do it, which would make the operator's cognitive load unbearably high. Our final lesson is that the robot should be capable of at least some degree of introspection. For, even if the operator does not know the exact set of goals the robot can achieve that set of goals can be discovered through introspection, i.e., by asking the robot what it can do.

Our current implementation has three limitations which we intend to address in our future work. First, our implementation does not handle negation. The robot has no way of interpreting commands such as "Do not do that!". Chapman's system (Chapman 1991) deals with negation by setting on and off bits in the appropriate linguistic buffers.

Second, unlike the system proposed by Fitzgerald and Firby (2000), our system does not deal with anaphora resolution. In other words, the operator cannot refer to objects with pronouns like "it" or "that" and expect the robot to figure out the correct reference. Third, our implementation is redundant in that the robot maintains two distinct types of memory: the semantic network in the deliberation tier and the RAP memory in the execution tier. While we cannot offer any cognitive argument against this arrangement, it seems redundant on software engineering grounds. Since the semantic network formalism allows one to express everything expressible in the RAP memory formalism (but not the other way around!), it makes sense to have the RAP system use the semantic network as its own memory, too. Finally, learning in our robot is also limited in that the only kind of learning the robot does is instruction-based.

Conclusions

We presented an approach to hands-free interaction with a robot through spoken dialog. Our approach is based on passive knowledge rarefication through goal disambiguation, a technique that allows the robot to refine and acquire knowledge through spoken dialog with a human operator. We discussed how our approach achieves four types of human-robot interaction: command, elaboration, introspection, and instruction-based learning. We showed how the approach was implemented on an autonomous robot.

References

Bonasso, R.P.; Firby, R.J.; Gat, E.; Kortenkamp, D.; and Slack, M. 1997. A Proven Three-tiered Architecture for Programming Autonomous Robots. *Journal of Experimental and Theoretical Artificial Intelligence* 9: 171-215.

Bookstein, A.; Kulyukin, V.; and Raita, T. 2002. Generalized Hamming Distance. *Information Retrieval* 5:353-375.

Chapman, D. 1991. *Vision, Instruction, and Action*. Cambridge, Mass.: MIT Press.

Cheng, G., and Zelinsky, A. 2001. Supervised Autonomy: A Framework for Human-Robot Systems Development. *Autonomous Robots* 10(3):251-266.

Firby, R. J.; Prokopowicz, P.; and Swain, M. 1995. Collecting Trash: A Test of Purposive Vision. In Proceedings of the Workshop on Vision for Robotics. International Joint Conference on Artificial Intelligence, Inc.

Fitzgerald, W., and Firby, R. J. 2000. Dialogue Systems Require a Reactive Task Architecture. In *Proceedings of the AAAI Spring Symposium*. Stanford, Calif.: AAAI Press.

Fong, T., and Thorpe, C. 2001. Vehicle Teleoperation Interfaces. *Autonomous Robots* 11:9-18.

Horswill, I. 1995. Integrating Vision and Natural Language without Central Models. In *Proceedings of the AAAI Fall Symposium on Embodied Language and Action*. Stanford, Calif.: AAAI Press.

Kulyukin, V., and Morley, N. 2002. Integrated Object Recognition in the Three-Tiered Robot Architecture. In *Proceedings of the 2002 International Conference on Artificial Intelligence*. Las Vegas, Nev.: CSREA Press.

Kulyukin, V., and Settle, A. 2001. Ranked Retrieval with Semantic Networks and Vector Spaces. *Journal of the American Society for Information Science and Technology* 52(14):1224-1233.

Kulyukin, V., and Steele, A. 2002. Instruction and Action in the Three-Tiered Robot Architecture. In *Proceedings of the 2002 International Symposium on Robotics and Automation.* Toluco, Mexico: IEEE Press.

Martin, C. 1993. Direct Memory Access Parsing, Technical Report, CS93-07, Dept. of Computer Science, The University of Chicago.

Nilsson, N. 1969. A Mobile Automaton: An Application of Artificial Intelligence Techniques. In Proceedings of the First International Joint Conference on Artificial Intelligence. International Joint Conference on Artificial Intelligence, Inc.

Rich, C.; Sidner, C.; and Lesh, N. 2001. COLLAGEN: Applying Collaborative Discourse Theory to Human-Computer Interaction. *AI Magazine* 22(4):15-25.

Torrance, M. 1994. Natural Communication with Robots. Masters thesis, Dept. of Computer Science and Electrical Engineering, MIT.

Winograd, T. 1972. *Understanding Natural Language*. New York: Academic Press.