Speech and Action: Integration of Action and Language for Mobile Robots

Timothy Brick
Department of Computer Science and Engineering
University of Notre Dame
Notre Dame, IN 46556, USA
tbrick@nd.edu

Paul Schermerhorn and Matthias Scheutz
Cognitive Science
Indiana University
Bloomington, IN 47401, USA
{pscherme,mscheutz}@indiana.edu

Abstract— **We describe the tight integration of incremental natural language understanding, goal management, and action processing in a complex robotic architecture, which is required for natural interactions between robots and humans. Specifically, the natural language components need to process utterances while they are still spoken to be able to initiate feedback actions in a timely fashion, while the action manager might need information at various points during action execution that must be obtained from humans. We argue that a finer-grained integration provides much more natural human-robot interactions and much more reasonable multitasking.**

I. Introduction

Robot architectures are becoming so complex that, from a software engineering point of view, it is virtually a necessity to design and build them modularly. However, the strict segregation of functionality is not without cost. In humans, for example, action management and natural language processing (NLP) are tightly integrated. Among other advantages, this integration allows humans to respond to spoken sentences (e.g., to nod in agreement or to begin executing an order) even before they are complete; speech processing continues as the response is carried out. This ability can yield important advantages, as we argue below, and, moreover, humans are accustomed to such responses and will expect them when interacting with robotic agents. Yet, without some integration of NLP and action management, such responses can be difficult to achieve systematically.

We believe that, although it is important to maintain the advantages of modular system design (the extreme instance of integration being a monolithic system), a genuine integration of language and action processing is highly desirable. Beyond the benefits to human-robot interaction, however, there are other important reasons why one might be interested in the integration of language and action. The *genuine integration* of language and action processing at a fine-grained time scale allows for rich interactions between them at various levels, while maintaining the benefits of modular design. Practically speaking, this can improve performance on tasks that mobile robots are increasingly being asked to perform (e.g., planetary exploration, as described below). We describe the integration between NLP and action processing in our system, and show how this integration can lead to improved outcomes in a collaborative human-robot task.

The rest of the paper is laid out as follows. We begin with a short discussion of background and prior work, and then an overview of the various parts of our proposed system and the mechanisms used for integration. We demonstrate the utility of these mechanisms in a sample scenario (experiments carried out in our lab with a robot and human subjects), and discuss briefly the utility of integration in other scenarios. We end with our conclusions and suggestions for future work.

II. Background and Related Work

Language and action processing are tightly integrated in humans. Recent evidence from neuroscience, in particular, suggests that areas (like Broca's area) previously assumed to be responsible only for language processing are also involved in action processing [9]. This tight integration of language and action allows human speakers to control various body parts (like their heads and hands) while speaking, tightly synchronizing the movements of their bodies with production of their speech. Similarly, it allows human listeners to perform a variety of actions while still processing a sentence (e.g., providing feedback to the speaker using nodding, shift in eye gaze, changes in facial expressions, etc.).

Autonomous mobile robots that interact with humans using natural language are thus faced with a fundamental challenge: if their interactions with humans are supposed to seem *natural* (i.e., human-like), then they will have to be sensitive to the styles and cadences of human language interactions, including the perception and production of concomitant gestures and other actions while processing utterances (both on the perception and production side).

Current architectures in robotics, however, typically treat language processing and action control as functionally separate modules where information is exchanged only in the form of block commands (generally the semantic meaning of an entire utterance), and often times in a unidirectional way (i.e., from language processing to action execution, but not vice versa). More importantly, language processing often proceeds in stages that deal with different aspects of an utterance (from the phonological stage in which sounds are translated into words, to the syntactic stage in which the grammatical structure of an utterance is isolated, to the semantic interpretation stage in which meanings are determined). Clearly, such a stage-based approach to natural language processing puts severe *a priori* restrictions on the kinds of integration possible between language processing and action management. For example, a listener's head nod (expressing the listener's understanding of or agreement with the meaning of the speaker's utterance so far) provides feedback to the speaker that can shape the way the speaker continues the sentence. A stage-based system that needs to finish processing the auditory signal before it is able to analyze its meaning is unable, in principle, to provide such feedback. Conversely, an action system that is carrying out a command might profit from being able to generate a natural language question if it determines that information about some aspects of the command are missing. Further, it may be possible to receive constraining feedback while continuing to perform actions that will allow it to complete the command (e.g., if grasping the only visible object of a particular type reveals another object of that type that was previously occluded and could be another candidate for the grasping action).

The tight integration of action and language is most often attempted in the pursuit of integrating language and vision. Used in this way, action and visual processes are used to constrain and enhance linguistic comprehension, learning, and disambiguation. [6], for example, integrates action as a support for visual learning from language. Other systems are not designed with the intent of performing multipart action sequences, but rather for other specific intents. For example, [2] integrate action for the purposes of emulating neurological data, here the neurological connectivity of the visuomotor system. [5] integrate action and language with a visual system in a different way; building internal representations for real and imagined worlds for motor tasks and answering "what-if" type questions. Again, these systems integrate language and action for specific intents distinct from the execution of multipart action sequences.

Fitzgerald, Firby, and Martin use action to constrain language by utilizing a shared conceptual memory to provide action scripts, state data, and "facts" for language disambiguation (e.g., see [3]). For example, the violation of preconditions of actions or impossible actions might be used to determine that certain interpretations are not possible, or the current state of the robot might make "move the lever" refer to a single specific lever, even if multiple levers exist.

We expand on all of these works by adding three necessary components to the integration of language and action. The ability to provide feedback appropriate to the understanding of the system during a speaker's utterance provides more natural conversation. The ability of action to begin execution of partially specified actions extends this by demonstrating the understanding of the system through initial actions. The online modification of running action scripts allows more complex interactions under time and processing constraints, and permits simultaneous execution of actions and clarification of misspecified or underspecified referents. We discuss these three contributions in greater detail in Section IV-D.

III. Component Parts

Our overall architecture is based on the distributed integrated affect, reflection, and cognition (*DIARC*) architecture (discussed in [7]). The architecture is currently implemented on an ActivMedia Peoplebot (P2DXE) with two Unibrain firewire cameras mounted on a Directed Perception pan-tilt unit, a Sick laser range finder, a Voice Tracker Array Microphone, and two speakers.

The architecture is used as a testbed for a variety of investigations of human-robot interaction, ranging from simple dialogue-like interactions for survey administration to more complex interactions involving a variety of subsystems such as vision, attention, action sequencing, affect, and dialogue.

Phonetic tokenization and resolution of speech is performed by the Sphinx-4 language recognition system.[1] Speech synthesis, when necessary, is performed by the Festival speech production system, described in [8].

A. Action Management

Action selection and management is accomplished by a goal-based system (described in [7]) in which the priority of each goal determines whether the resources required to achieve that goal are allocated to it. Procedural knowledge in the form of action scripts serves as the basis for action sequencing. When the architecture is presented with a goal, a script that can accomplish that goal is retrieved from long-term memory. In many cases, these scripted action sequences are general descriptions of how a task can be accomplished, and certain variables in the script must be bound to values relevant to a specific context (e.g., a script describing how to take an item to a location would require as parameters at least the item to be transported and the destination). Action descriptions in the script are then translated into concrete actions (i.e., calls to other DIARC components, such as motion, speech recognition, and speech production) by an action interpreter subcomponent.

When multiple goals are being serviced concurrently, all associated scripts are able to make progress toward their goals, so long as no conflicts arise over resources. So, for example, it would be possible for the robot to pursue the goal of moving to some other location while at the same time pursuing the goal of carrying on a direct conversation with a person accompanying it to the new location. If, however, their paths did not coincide, a conflict would arise between the travel goal's need to issue motor commands to move the robot toward its destination and the conversation goal's need to issue motor commands to halt the robot to be able to continue the conversation.

In these cases, the *importance* of each goal is combined with the goal's *urgency* to determine the goal's *priority*. A goal's importance (i) is a measure of its utility (benefit b minus cost c), possibly scaled by the influence of affective states (a_p and a_n, positive and negative affect, respectively)

[1]More information on Sphinx-4 is available from the project website at http://fife.speech.cs.cmu.edu/sphinx/

in the architecture:

$$i = (1 + a_p) \cdot b - (1 + a_n) \cdot c. \quad (1)$$

The urgency (u) of a goal is a function of the total time allocated to that goal (t_a) and the time elapsed since it was instantiated (t_e), possibly bounded by minimum and maximum urgency values (u_{min} and u_{max}, respectively):

$$u = \frac{t_e}{t_a} \cdot (u_{max} - u_{min}) + u_{min}. \quad (2)$$

The product of the goal's importance and urgency constitute its priority:

$$p = i \cdot u. \quad (3)$$

When a conflict arises over a resource, the goal with the higher priority obtains control of the resource, until the goal is achieved or it is preempted by a yet higher-priority goal (maybe even the same goal that formerly had a lower priority if, for example, its urgency increases at rapid rate).

Having a high priority value is not sufficient to ensure that a goal's actions can execute. Other preconditions must also be met, including the acquisition of all necessary information for determining the parameters of an action. The main target of DIARC is to explore aspects of human-robot interaction, with the primary means of communication being verbal. Thus, interaction with the discourse engine is both frequent and crucial to goal achievement. The action management component includes facilities to obtain needed information from the natural language system for script execution, as well as facilities to allow discourse to notify the goal manager when the requested information is available (and, potentially, to provide updates as new information is acquired).

A straightforward example is the "follow orders" goal. In this case, the information required includes the action to be executed (e.g., "move"), but may also include parameters or other constraining information (e.g., "move to the home base," or "move forward until the path is blocked"). In many cases, it is impossible (or impractical) for the robot to proceed without having the parameters fully specified (e.g., it may not make sense to commence the "move to" goal before finding out the destination, as any actions taken could place the robot further from the goal than when it started). However, there are other times when acquiring the additional information can be postponed until it is needed. In either case, the action manager requests a binding of type "action" from the discourse engine and waits to be notified of the response. When the binding arrives, the action interpreter attempts to execute the script elements corresponding to the command. If the command is fully specified (e.g., "stop"), the script can execute. If the command is missing crucial elements, as in the "move to" example above, the action interpreter must postpone the action until they are provided. If, however, the missing elements are not immediately needed, it is possible for the robot to make some progress on the goal before progress is blocked by the missing information. The script begins execution while leaving open a "channel" via which discourse can update the parameter list. If the missing information is provided before it is needed, the script proceeds exactly as it would have had the information been present from the start. Otherwise the script will block at the point where the new binding is required and will generate another binding request for the discourse engine to acquire the missing information.

B. Natural Language Processing

Our incremental discourse engine, called TIDE, is designed to combat the problems of noise and processing time by performing simultaneous incremental syntactic and semantic processing of an utterance as it is being spoken. The incremental nature of the processing system makes it robust to misunderstood or incomplete speech, and its ability to semantically process statements as they are being spoken allows it to provide meaningful "back-channel" feedback before the utterance is completed. Details on TIDE's syntactic and semantic engine (called RISE), its processing model, algorithms, and reanalysis procedures (in the case of a garden-path), are given in [1].

Of primary importance to this paper, TIDE incrementally interprets utterances as they are being spoken. As each word of the discourse is interpreted, meaningful partial interpretations are passed to the action interpreter, marked with an indicator of the completeness of the utterance. Simultaneously, so-called "back-channel" feedback indicating the robot's state of understanding (say, by nodding) or indicating that reference has been established (say, by looking at the object of reference) can also be requested. While any model of feedback generation could be used to decide which type of feedback to perform when, it is the basic theoretical capability and not the model which is important to this paper.

In order to ensure that context is maintained across conversational topics within a conversation, TIDE extends RISE through the addition of two persistent discourse structures: a *discourse context* and an *interaction template*. The *discourse context* represents the shared discourse history from the current interaction (and potentially from the contexts of past interactions). Its primary function is the resolution of anaphoric expressions in conversation and the determination of bindings for action scripts where they are not explicitly stated. Discourse contexts in TIDE use an annotated stack for maintenance of past referents, and fill in the most likely recent referent that meets syntactic and semantic guidelines, similar to the system proposed by [4]. References determined by grammatical conventions (such as the implied actor in a command) are maintained separately.

The *interaction template* provides a boilerplate for a subject of discourse, and contains syntactic/semantic structures for words and their associated meanings.[2] This prevents confusion from sentences such as "This group has no identity," the words of which have a very different meaning in a mathematical context than in a sociological context

[2]In this context, the 'meaning' of a word is the internal representation associated with it. For example, the 'meaning' of the command "move to the left" is the action script *startMoveLeft:rudy* that will cause the robot to move.

(suggested in [10]). The differences between these contexts are determined by the context maintained in the *interaction*.

As a sentence is interpreted, semantic/syntactic structures are pulled from the interaction template and used in a constraint-propagation system to incrementally determine the meaning of an utterance. In the case of commands, an utterance is translated into the relevant action script with appropriate variable bindings. Some of these bindings are evident from the utterance (the actor in the case of a command given to the robot, for example, is the robot itself), some need to be determined from visual context (as described in [1]), and some need to be determined from the discourse context, as described above.

In the event that a necessary binding cannot be found, a *subinteraction* is added to the interaction stack and a clarifying question or set of questions is generated. A subinteraction adds the context of a limited interaction meant to resolve the missing bindings. For example, if the robot were asked to turn, it could generate the partial action script "startTurn?direction". Because this is not a useful partial command, TIDE would ask "Which direction?" and generate a subinteraction capable of understanding partial (and non-grammatical) answers to that question. If the response came back, "left", the response would be appropriately inserted into the action script, and the "startTurnLeft" action begun. The subinteraction would then be removed from the interaction stack, since responses like "left" have no meaningful syntactic nor semantic interpretation in the absence of an appropriate context. Subinteractions do not replace the current interaction, but rather temporarily add to it, so that if the human were to answer the question with, "No, go straight", the utterance could be processed normally.

This separation of contextual information into "meanings" (that is, *interaction templates*) and "memory" (in the form of *discourse contexts*) allows a single conversation with a single individual to maintain past referents despite changing topics and modes of conversation (say, from a technical discussion of mathematics to a more general discussion of sociology).

C. Integration Mechanisms

Communication between the action manager and TIDE takes two main forms: the action manager can request data of a specific type (e.g., "action" or "destination") from discourse, and TIDE can set those bindings in the action manager. In the general case, the action manager determines that some piece of information is missing (i.e, there is some variable in the script representation that has not been bound to a value) that it needs in order for the current goal to progress. In many cases, the robot is able to fill in the missing variable bindings directly by examining its environment (e.g., by having the vision system identify an object, or by having the localization system determine the current position). In some cases, however, it is necessary to get the needed information from a human interlocutor. The action manager requests a binding of the appropriate type from the discourse engine, which uses the type information, along with other information in the current context, to construct an appropriate query. When TIDE has acquired the requested information, it fills in the binding in the action manager via a "set" call.

In the example used below, the action manager requests a binding of type "action," indicating that it is ready to accept commands from a human partner. As TIDE begins understanding an incoming command, it passes partial interpretations to the action manager so that the action can be started as soon as possible. Each command passed in this way is assigned a unique identifier so that later interpretations of the same command are interpreted as refinements on the initial interpretation rather than as separate commands.

In the event that an utterance completes without filling in all the variable bindings required by the appropriate action script, TIDE will search the discourse context for an appropriate binding. If such a binding does not exist in the discourse context, TIDE will generate a question and an appropriate subinteraction to handle the result. Meanwhile, TIDE sends the command to the action manager, leaving empty variables in the place of the missing parameters. The action manager can then begin to execute the action sequence to the extent possible without the missing parameters. When the human's response has been interpreted and new information is available, TIDE updates the running action script, binding the new value to the (empty) parameter variable.

In the event that no further progress can be made by the action manager without bindings for missing parameters, it can generate explicit binding requests for TIDE. As the discourse engine does not have full access to the procedural knowledge encoded in the action scripts, it is not always possible for it to anticipate which variable bindings will be needed soonest. Explicit binding requests from the action manager forcibly prioritize the bindings currently most crucial to making progress toward the current goal.

IV. Examples of Behavior

Some benefits of the integration mechanisms described above are demonstrated here using a hypothetical planetary exploration task. The following scenario is a variant on a task used many times in experiments with human subjects and the Peoplebot described above to study various aspects of human-robot interaction. The human and the robot form a team whose goal is to transmit data collected on the surface to an orbiting satellite. The signal from the satellite is only strong enough for transmission at certain locations on the surface. The robot serves both as transmission hardware and data storage mechanism (so data need not be transferred to the robot before sending), and it keeps track of the nearest detected transmission point. It is assumed that satellite time is expensive, so transmissions must be kept to a minimum.

The data has been separated into several different types, so that only the types that are needed must be uploaded to the satellite. When a human gives the order to upload a given type of data, the robot must move to the nearest transmission point and then transmit the appropriate data to the satellite. In the examples described here, the human initially does not specify which type of data, so this information must be determined by continued discourse. We present below three system traces of interactions with the robot demonstrating three different levels of integration.[3]

We have previously used the DIARC architecture and variants on our exploration task in several experiments, both with and without humans. A previous version of the architecture, action manager, and natural language processor was presented at AAAI 2006, where it received an award for natural language processing and action execution [7].

A. Unintegrated: Collect and pass on

Perhaps the most common standard of integration between components, the first level of integration requires very little communication between the discreet modules. The discourse engine knows what parameters must be passed for each action script, and fills them before sending the completed command to the action manager.

```
Brady:     Upload the data.
<Discourse Interaction - Explorer got
        command:[upload:rudy:?datatype]>
Rudy:      Which data type?
Brady:     Images.
<Discourse subinteraction - datatypeFinder
        datatype:[image] for
        command:[upload:rudy:image]>
{Action Manager got binding:
        command:[upload:rudy:image]}
{Action Manager begins
        action:[upload:rudy:image]}
Rudy begins to move.
...
Rudy reaches transmission point.
Rudy transmits image data.
```

Notice that no actions are begun before the entire exchange is completed. The human issues the command with incomplete data, and the robot asks clarifying questions until all the appropriate bindings are filled. The action manager then receives a complete action specification, which it then executes. While this strategy takes little effort to implement, and provides a minimally useful level of interaction, valuable time is wasted while the robot waits for the command before moving to the transmission point.

B. Partially integrated: Ask and ye shall receive

With partial integration, more interaction between the systems is possible (and likely). Partial integration allows the discourse engine to respond to binding requests with incomplete matches, such as action specifications with unbound parameters. When action execution stalls because of an unbound variable, the action manager requests a new binding to fill the parameter.

Because the discourse engine only responds to explicit requests from the action manager, it will send incomplete action specifications when no bindings are available in the current context. When action subsequently requests a binding of type *datatype*, discourse spawns a subinteraction and asks the human partner for more information.

```
Brady:     Upload the data.
<Discourse Interaction - Explorer got
        command:[upload:rudy:?datatype]>
{Action Manager begins
action:[upload:rudy:?datatype]}
Rudy begins to move.
...
Rudy reaches transmission point.
{Action Manager requests binding of
        type datatype from Discourse}
<Discourse processing binding request>
Rudy:      Which data type?
Brady:     Thermal data.
<Discourse subinteraction - datatypeFinder
        datatype:[thermal]>
{Action Manager got binding:
        datatype:[thermal]}
Rudy transmits thermal data.
```

Notice that in this scenario, the discourse engine sends the *upload* command to action with the *datatype* unbound. When action reaches the point where it requires that binding, it requests the type from the discourse engine, which generates a subinteraction as before.

Although this approach does allow the robot to make progress toward achieving its goals even when complete action specifications are unavailable, there is the potential for confusion or even failure, as the robot may not detect the missing parameter until much later (e.g., the transmission point may be far away, and the human partner may not accompany the robot there). A better solution would recognize the missing parameters and proactively attempt to bind them, without impeding actions that do not rely on missing values.

C. Fully Integrated: Modify on the fly

In our final level of integration, the discourse engine has the ability to modify the parameters of an executing action without interruption. This allows the discourse engine to continue the conversation and request the bindings needed for an action while the action manager begins execution of the action script.

```
Brady:     Upload the data.
<Discourse Interaction - Explorer got
        command:[upload:rudy:?datatype]>
{Action Manager got binding:
        command:[upload:rudy:?datatype]}
{Action Manager begins
        action:[upload:rudy:?datatype]}
{Action Manager step 1:
        action:[move:Rudy:transmitPoint]}
Rudy begins to move.
<Discourse created subinteraction:
        datatypeFinder>
Rudy:      Which data type?
Brady:     The maps.
<Discourse subinteraction - datatypeFinder
        datatype:[map]>
{Action Manager got binding updates:
        action:[upload:rudy:map]}
...
Rudy reaches transmission point.
{Action Manager step 2:
        action:[transmit:rudy:map]}
Rudy transmits map data.
```

Notice that the execution of the partial command *upload:rudy:?datatype* begins by the time the utterance is complete, while conversation continues with the human partner. As more information becomes available to the robot,

[3]These traces are intended to demonstrate the functioning of the system, and as such have been edited, annotated, and reformatted.

the discourse engine fills in the missing bindings in the executing action context so that the action can complete without interruption.

This approach solves the problems of both the above approaches simultaneously, as well as allowing additional capabilities, such as back-channel feedback, described above.

D. Discussion

The experimental runs presented above highlight the shortcomings of fully modular NLP and action management subsystems and demonstrate some advantages provided by the integration capabilities of TIDE and DIARC. Incremental speech processing allows on-line, meaningful nonverbal feedback concurrent with further speech processing. The system is able to initiate actions that are not fully specified and allows on-line modification of executing scripts (e.g., to "fill in" missing parameters for a running script). The example traces presented above are intentionally simple, to make it easy to distinguish between them. However, the capabilities presented are applicable to many, more realistic scenarios, each of which can benefit greatly from integration of the kind we propose here.

Consider the situation of a robot separated from its operators by a long distance (say, from a rover on Mars to a base on Earth) in a similar scenario to the one detailed above. A robot which is able to initiate underspecified actions could begin to locate an appropriate transmission point while waiting (approx. forty minutes) for the response from Earth about which data to transmit. If the transmission window is limited, the time savings could mean the difference between a successful transmission and the need to wait another day to upload the data.

Consider also cases in which a complex action sequence is currently being carried out by the robot, and some part of the order needs to be amended (e.g., because it was incorrect or because new information became available). For example, if the robot began to move towards the transmission point and then heard "no, transmit the mineral data instead", the ability to alter running scripts would allow the robot to correct its action (and transmit the appropriate type of data) without incurring the delay of canceling the current action and replanning an entire new action.

Finally, consider a robot with limited computation allotted for planning. The integration capabilities of the action manager are not NLP-specific; other subsystems that generate action specifications can also take advantage of the mechanisms described above. For example, the ability to execute partial plans could allow the robot to design a "rough" plan and begin executing it while simultaneously constructing a refined plan to be substituted in when ready.

V. Conclusions and Future Directions

Beginning with the observation that human action selection and language processing systems are tightly coupled, we argue that the lack of genuine integration between language processing and action execution systems in robotic systems makes many aspects of human natural language interaction difficult, if not impossible. A finer-grained integration of these systems can have a dramatic effect on their ability to perform certain tasks, and in some cases can provide performance advantages to autonomous robot systems. We presented a system that implements this finer integration, and showed by example how it is able to perform these tasks in a way that is both more optimal and more "human."

Future directions involve the inclusion of more metadata in the interaction of the two systems so that requests from the action management system can be more gracefully requested in natural language. A predictive system capable of anticipating new needs before and as they arise and request expected bindings before they are needed will improve system reactivity. The increased planning ability this provides will simultaneously add to the robot's ability to deal with strange and unexpected circumstances.

To completely evaluate the online utility of the system, a human study comparing task performance on an explore-and-report task such as the one examined in Section IV could be performed. It is expected that the theoretical results reported in this paper would accumulate into a noticeable improvement in task performance time.

As robotic systems become more sophisticated, the integration between their component parts becomes more and more crucial to their everyday operation. A sensitivity to the way that humans interact, and a fine-grained integration between modules of such complicated products will ultimately lead to robotic systems that are more capable of simultaneously acting optimally and interacting naturally with humans.

References

[1] T. Brick and M. Scheutz. Incremental natural language processing for HRI. In *Proceedings of the 2nd ACM International Conference on Human-Robot Interaction*, page forthcoming, 2007.

[2] Rebecca Fay, Ulrich Kaufmann, Heiner Markert, and Gunther Palm. Integrating object recognition, visual attention, language and action processing on a robot in a neurobiologically plausible associative architecture. In *Proceedings of the AI-Workship on NeuroBotics*, Ulm, Germany, 2004.

[3] Will Fitzgerald and R. James Firby. The dynamic predictive memory architecture: Integrating language with task execution. In *Proceedings of the IEEE Symposia on Intelligence and Systems*, Washington, D.C, 1998.

[4] Barbara J. Grosz and Candace L. Sidner. Attention, intentions, and the structure of discourse. *Comp. Ling.*, 12(3):175–204, 1986.

[5] Nikolaos Mavridis and Deb Roy. Grounded situation models for robots: Bridging language, perception, and action. In *AAAI-05 Workshop on Modular Construction of Human-Like Intelligence*, 2005.

[6] P. McGuire, J. Fritsch, J. J. Steil, F. Rothling, G. A. Fink, S. Wachsmuth, G Sagerer, and H. Ritter. Multi-modal human-machine communivation for instructing robot grasping tasks. In *IROS 2002*, Lausanne, Switzerland, 2002.

[7] P. Schermerhorn, J. Kramer, T. Brick, D. Anderson, A. Dingler, and M. Scheutz. Diarc: A testbed for natural human-robot interactions. In *Proceedings of AAAI 2006 Robot Workshop*, page forthcoming, 2006.

[8] Paul A Taylor, Alan Black, and Richard Caley. The architecture of the festival speech synthesis system. In *The Third ESCA Workshop in Speech Synthesis*, Jenolan Caves, Australia, 1998.

[9] Roel M. Willems, Asli Özyürek, and Peter Hagoort. When language meets action: The neural integration of gesture and speech. *Cerebral Cortex*, 2006.

[10] Terry Winograd. *Understanding Natural Language*. Academic Press, 1972.