Following Assembly Plans in Cooperative, Task-Based Human-Robot Dialogue

Mary Ellen Foster
Informatik VI: Robotics and Embedded Systems
Technische Universität München
foster@in.tum.de

Colin Matheson
School of Informatics
University of Edinburgh
colin@inf.ed.ac.uk

Abstract

The JAST dialogue system allows a human and a robot to jointly assemble construction toys on a common work area. Supporting this type of dialogue requires that the system have a representation of assembly plans that permits it both to discuss the details of the plan and to monitor its execution. We present a conceptual representation of assembly plans based on AND/OR graphs, and then describe how the dialogue manager uses these plans as the basis for a range of strategies for jointly carrying out the plans with the user.

1 Introduction

An increasing number of interactive systems are addressing the task of supporting intelligent cooperation with a human partner, where both partners work together to achieve a mutual task. This type of task-based collaboration is particularly relevant for robots, which are able to sense and perform actions in the physical world and can often be treated as full team members (Breazeal et al., 2004; Fong et al., 2005). For an artificial system to be able to work together with a human on such a task, the details of the task must be represented in such a way that the system can both follow the task progress and participate in discussing the details of the task execution.

In this paper, we present the JAST human-robot dialogue system, which allows the user to cooperate with the robot in assembling wooden construction toys. Assembly plans are represented as AND/OR graphs (Homem de Mello and Sanderson, 1990), which is the standard mechanism for representing such plans in autonomous robot assembly. This representation allows the dialogue manager to access the current steps in the plan and to update the state of the world following user actions. The dialogue manager implements two strategies for explaining a plan to the user, one that traverses the plan in a depth-first way, naming objects after they are complete, and another that names and describes the objects top-down.

The interactions supported by the JAST system is quite similar to the 'Max' virtual communicator system developed at the University of Bielefeld (Kopp et al., 2003; Rickheit and Wachsmuth, 2006). However, the mechanisms underlying the interactions are different: while the core of Max is a cognitively-motivated agent architecture, JAST uses a dialogue manager based on the information-state update paradigm. Our implementation also shares some features with Blaylock and Allen (2005)'s *collaborative problem-solving* (CPS) model of dialogue. That model divides the problem-solving process into three general phases: determining objectives, determining and instantiating recipes, and executing recipes and monitoring success. While we do not employ the full formal structure of the CPS model, the JAST system views collaborative dialogue in a similar way. A similar link between domain plans and dialogue strategies is also used in the LeActiveMath mathematics tutorial dialogue system (Callaway et al., 2006) to allow the system to describe and cooperatively follow plans drawn from a domain reasoner and to give context-specific hints to guide a learner through the graph of a solution.

2 The JAST human-robot dialogue system

The overall goal of the JAST project ('**J**oint **A**ction **S**cience and **T**echnology') is to investigate the cognitive and communicative aspects of jointly-acting

Proceedings of the 12th Workshop on the Semantics and Pragmatics of Dialogue, June 2–4, 2008, London, U.K.

Figure 1: The JAST dialogue robot

agents, both human and artificial. The JAST human-robot dialogue system (Rickert et al., 2007) is designed as a platform to integrate the project's empirical findings on cognition and dialogue with its work on autonomous robots, by supporting multimodal human-robot collaboration on a joint construction task. The user and the robot jointly assemble wooden construction toys on a common workspace, coordinating their actions through speech, gestures, and facial displays.

The robot (Figure 1) consists of a pair of mechanical arms with grippers, mounted in a position to resemble human arms, and an animatronic talking head able to produce facial expressions, rigid head motion, and lip-synchronised synthesised speech. The input channels consist of speech recognition, object recognition, robot sensors, and face tracking; the outputs include synthesised speech, head motions, and manipulator actions.

In the current version of the system, the robot is able to manipulate objects in the workspace and to perform simple assembly tasks. The primary form of interaction with the current system is one in which the robot instructs the user on building a particular compound object, explaining the necessary assembly steps and retrieving pieces as required; at the end of the paper, we discuss extensions to this scenario. To make joint action essential to the assembly task, the workspace is divided into two areas: one belonging to the robot and one to the human. The pieces necessary for building a desired assembly are distributed across these areas so that neither of the agents is able to reach all of the required components and must rely on the partner to retrieve them.

3 Representing assembly plans

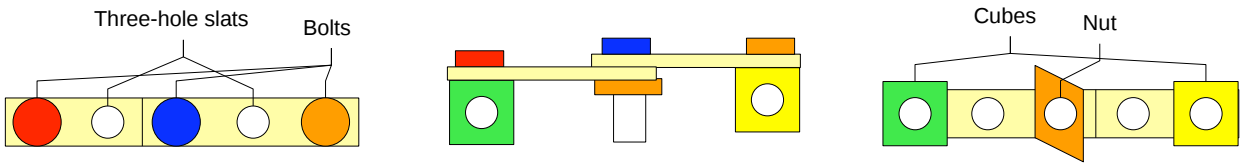
Like several previous interactive systems designed to support (physical or virtual) joint assembly (e.g., Knoll, 2003; Rickheit and Wachsmuth, 2006), dialogues in JAST are based around assembling *Baufix* wooden construction toys. The following are the basic components that are available:

- Threaded **Bolts** of varying lengths and colours;
- **Cubes** of varying colours, with four threaded holes and two unthreaded holes;
- **Nuts** with a single threaded hole; and
- **Slats** with three, five, or seven unthreaded holes

For the remainder of this paper, we will consider the sample object shown in Figure 2, which we will call a 'bridge'. This object consists of two small (three-hole) slats, connected end-to-end using a blue bolt and a nut, with a cube connected to the other end of each slat. Some of the sub-components also have names: the slat+cube combination on the left of Figure 2 is called the 'front', while the combination on the right is called the 'back'.

Even for this fairly simple object, there are a number of different possible assembly sequences: the slats may be joined together at any point, and the two cubes can also be attached in any order. There is some symmetry in the plan: for example, the two slats are interchangeable, and it is not important which end of a slat or which hole in a cube is used. However, there are also geometric relationships among the pieces that must be respected, such as the fact that the bolts all go through the slats in the same direction.

In this section, we present the assembly-plan representation used in JAST, which captures all of these features. We begin by describing the representation of individual assembly steps and then show how those steps are combined to describe the full plan.



(a) Top view (b) Side view (c) Bottom view

Figure 2: Assembled object ('bridge')

3.1 Assembly steps

The first step in representing an assembly plan is to represent the individual assembly steps. In our system, assembly steps are represented in a domain-specific way, tailored to the types of objects that can be constructed from Baufix components. Following Sagerer et al. (2002)—who also represented Baufix assemblies for use in interactive assembly—we define an assembly step to consist of the following components:

- Exactly one bolt;
- Any number of unthreaded pieces to insert the bolt through (cubes, slats, or composite objects containing cubes or slats); and
- Exactly one threaded fastener to screw onto the bolt (a nut, a cube, or a composite object containing a cube).

In addition to the above features, an assembly-step description also includes details to ensure that the step is performed correctly. These details fall into three main classes:

- For any component that is a composite object, which piece of that object should be used;
- Which of the several holes in a slat should be used; and
- The direction of insertion or fastening.

Note that not all of these details are necessary for a single step: it does not matter which of the four threaded holes in a cube is used for an attachment operation, for instance, and the two end holes of a slat are also interchangeable, as are the two faces. However, when the same component is used in more than one assembly step—as in the sample object,

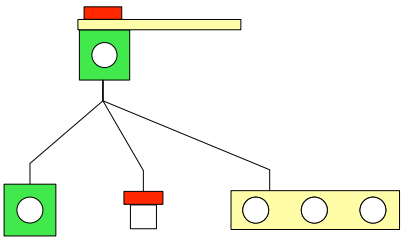


Figure 3: Single assembly step

Bolt $b_1(\text{bolt}, \text{small}, \text{red})$
Insert list $[i_1(\text{slat}, \text{three-hole})]$
Fastener $f_1(\text{cube}, \text{green})$
Details $[\text{hole}(i_1) = \textbf{Middle}, \text{direction}(f_1) = \textbf{South}, \text{direction}(i_1) = \textbf{South}]$

Figure 4: Symbolic description of the assembly step

where each of the two slats is used twice—it is important that all of those steps are performed based on the same frame of reference to ensure that the relative positions of the objects are correct. We therefore define a canonical orientation of an assembled object (as in Figure 2(b)).

Figure 3 illustrates the assembly step that creates the 'front' of the sample object: a red bolt is inserted from above through the end of a three-hole slat and is then screwed into a threaded hole of a green cube. Figure 4 gives a symbolic description of this step.

3.2 Assembly plans

Each possible assembly plan for an object is made up of a sequence of assembly steps, where a single object may have a number of such sequences. In autonomous assembly, the standard solution for representing such a set of assembly sequences is the *AND/OR graph*: a directed acyclic graph that decomposes a problem into two sets of nodes, AND nodes and OR nodes. An AND node is satisfied

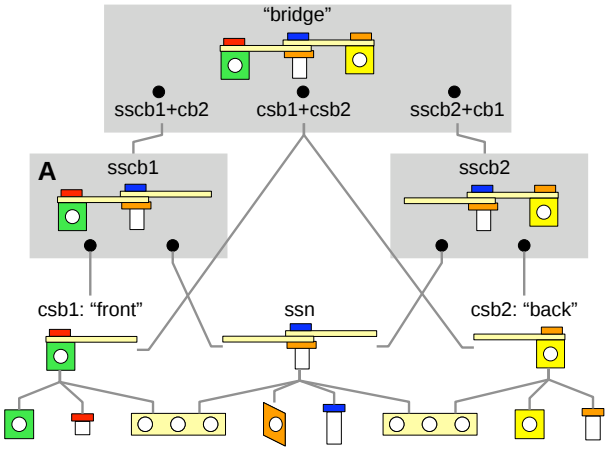


Figure 5: AND/OR graph structure for the sample object

only if all of its children are satisfied, while an OR node is satisfied when exactly one of its children is. This provides a natural representation for any problem that can be represented by decomposing a goal into subgoals, and was first proposed for robot assembly by Homem de Mello and Sanderson (1990).

In an assembly plan, an AND node corresponds to a single assembly step in which all of the children are combined to produce a more complex component. An OR node, on the other hand, corresponds to situations in which an assembly may be produced by different sequences of assembly operations; in this case, each child of the node corresponds to a different assembly sequence. An AND/OR graph provides a compact representation of all of the possible assembly sequences for an object; each individual sequence can be extracted by traversing the tree top-down, including all of the children of each AND node and exactly one child of each OR node.

Figure 5 shows the structure of the AND/OR graph for the sample object in Figure 2. Nodes with outgoing edges represent AND nodes, while nodes highlighted with a grey background are OR nodes. The leaf nodes in the tree correspond to the individual pieces required to build the sample object, while each internal node corresponds to a subassembly. For example, the subtree rooted at the OR node marked **A** indicates that there are two different assembly sequences that can result in that component. The first, corresponding to the left child of **A**, involves first attaching the green cube to one end of the slat to make the 'front' and then attaching the other slat to the other end. The second sequence, corresponding to the right child of **A**, first creates the (unnamed) centre piece and then attaches the cube to the end.

Each internal node has a unique ID—for example, the node corresponding to the assembly operation from Figure 3 has ID *csb1*. Three of the nodes also have labels indicating that the corresponding sub-assembly has a name: the 'bridge', the 'front' and the 'back'.

Previous systems have also addressed the task of representing assemblies of Baufix-style objects. Brock (1993) represented components by their geometric properties and described assemblies in terms of hierarchical planning operators. This system had the goal of creating plans for a robot to autonomously assemble the components, with no user interaction. Sagerer et al. (2002) used a similar representation for assembly actions to the one described here, with the goal of recognising complex objects in an interactive human-robot scenario. This system did not represent full assembly plans, but rather structural descriptions sufficient for recognition. The representation for Baufix assemblies in for the Max virtual communicator Jung (2003) described them in terms of *ports* and *connections* of CAD-based parts with the goal of supporting assembly in virtual environments.

The JAST representation described in this section is most similar to that used by Sagerer et al. (2002), although they do not use AND/OR graphs to represent the assembly plans; the other representations concentrate more on detailed geometric features that are less relevant to the current scenario where the user is the primary agent for assembly operations.

4 Following an assembly plan in dialogue

Interactions in JAST are based around cooperatively carrying out assembly plans represented as described in the preceding section. As mentioned earlier, in the current scenario, the robot is aware of the target object and the full plan and instructs the user on carrying out the assembly, and the user learns to make particular sub-components along the way. In Section 5, we discuss possible extended interactions, but in this section we concentrate on the robot-as-instructor scenario. Excerpts from typical interac-

Depth-first	
SYSTEM[1]:	First we need to build a bridge. Okay?
USER[1]:	Okay
SYSTEM[2]:	[*picking up green cube*] Insert the red bolt into the end of a slat and fasten it with this cube.
USER[2]:	Okay
SYSTEM[3]:	Well done. You have completed the front. Now insert ....
Top-down	
SYSTEM[1]:	First we need to build a bridge. Okay?
USER[1]:	Okay
SYSTEM[2]:	To build a bridge we need to make a front and a back. To make a front, insert the red bolt ...

Figure 6: Example depth-first and top-down interactions

tions using two different explanation strategies are shown in Figure 6. In the remainder of this section, we describe how the components of the system work together to support such interactions.

The required knowledge is distributed across three main components of the system. The **task planner** stores and maintains the AND/OR graph corresponding to the current plan, updating it as appropriate based on information from other modules. The **object inventory** tracks the properties and locations of Baufix objects in the world, using information from the object-recognition system as well as the task planner. Finally, the **dialogue manager** (DM) receives a unified representation of user speech and actions from the input-processing components and selects appropriate system output based on the current state of the interaction and of the plan, along with the user's assumed knowledge. It also updates the state of other components based on the events in the dialogue.

The DM is based on the TrindiKit dialogue-management toolkit, which uses the information-state update approach to dialogue management (Traum and Larsson, 2003). The JAST information state (IS) includes data about the user's knowledge, the current step in the plan that is being executed, the history of steps that have been described to the user, and the history of the interaction. Figure 7 in Section 4.2 below contains an example IS and some further discussion.

Table 1: Initial object inventory

ID	Type	Properties	Location
1	Bolt	Color=Red	Table(User)
2	Bolt	Color=Orange	Table(Robot)
3	Bolt	Color=Blue	Table(Robot)
4	Cube	Color=Yellow	Table(Robot)
5	Cube	Color=Green	Table(Robot)
6	Cube	Color=Green	Table(User)
7	Nut	Color=Orange	Table(Robot)
8	Slat	Size=3-hole	Table(User)
9	Slat	Size=3-hole	Table(User)
10	Slat	Size=5-hole	Table(User)

4.1 Loading the plan

Before the first utterance in the dialogue excerpt, the system must select a target object to assemble. In our scenario, where the robot knows the plan and must instruct the user, the choice of target object is fixed, so the DM simply instructs the task planner to load the AND/OR graph for the target object from its library of fully-specified plans. At the moment, the task planner also selects a specific assembly sequence from the AND/OR graph at the point that the plan is loaded, favouring sequences that include more named sub-components (e.g., the 'front') to ensure that the user learns to build them.

4.2 Describing assembly steps to the user

Once the AND/OR graph has been loaded and a sequence selected, the DM must describe the assembly process to the user. The DM can proceed **depth-first**, describing plan steps and naming objects when they are complete, or it can work **top-down** and describe and name each step in advance; both of these strategies are illustrated in Figure 6. In both cases the actual path through the plan is the same, and the 'current state' of the dialogue as represented in the IS is a crucial component. A truncated example of an IS (relevant to either strategy) is contained in Figure 7 and shows the basic plan information and typical dialogue history (DH) and user knowledge (UK) representations. The names of the plan nodes (e.g., 'front' or 'bridge') are associated inside the planner with (language independent) concepts which the language generation system turns into lexical items in English or German; the DM only needs to know the plan node identifier. The DH contains an ordered

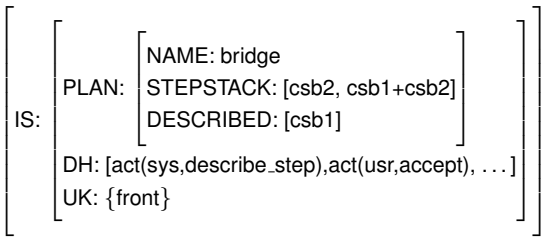


Figure 7: Part of the IS Structure

list of completed acts, and the user knowledge is represented as a set of 'known' object types. This set is maintained throughout the dialogue, so if we are constructing an object for the second time we can ask the user if they remember how to build it.

In operation, the DM first requests the children of the tree node corresponding to the step to be described. A check is carried out to determine whether all the objects mentioned in the step are either basic components (bolts, cubes) or known to the user; if not, the system picks the left corner child and iterates until such a node is found. If the depth-first strategy is being pursued, the DM proceeds without producing linguistic output until it reaches a node where everything necessary to build the object exists. In the top-down approach the system names each node and describes the general structure as it proceeds, whereas the depth-first strategy assumes that objects are built before they are named. The 'delayed naming' aspect of the depth-first approach is not, of course, necessary; it is perfectly possible to tell the user what is being constructed before it is described. However, the naming strategy is an aspect of dialogue that we would like to experiment with, and it seems less natural to combine top-down description with delayed naming.[1] As each step is completed, the DM sends the planner a 'step executed' message with the relevant node name, which updates the system state as described in Section 4.5. When the whole plan is complete, the system loads the next assembly plan or terminates.

[1]Top-down with delayed naming would suggest system utterances such as:

> Let's build a bridge. Insert a blue bolt through a green slat and fasten it with a yellow cube. This is a front. Now insert ...

From the perspective of Centering Theory (Grosz et al., 1995), the focus shifts are non-optimal.

```
<rst>
  <consequence id="id1">
   <item idref="id2" />
   <item idref="id3" />
  </consequence>
  <item id="id2" type="impersonal">
   <pred action="build" result="front" />
  </item>
  <join id="id3">
   <item idref="id4" />
   <item idref="id5" />
  </join>
  <item id="id4" type="imperative">
   <pred action="insert">
           .... contents of insert ....
   </pred>
  </item>
  <item id="id5" type="imperative">
   <pred action="fasten">
           .... contents of fasten ....
   </pred>
  </item>
 </rst>
```

Figure 8: An RST 'Consequence' Structure in XML

4.3 System Output

The DM builds XML structures containing RST-style representations (Mann and Thompson, 1988) to be passed on to the output planner and ultimately the language generator. The top-down strategy uses 'consequence' relations to link the actions being described and their results, as illustrated in Figure 8. The 'insert' and 'fasten' elements in the figure, which describe the objects, have been removed for brevity. The type attribute on item elements specifies the basic clause class; impersonal clauses such as 'to build a bridge', imperatives such as 'insert the bolt', and declaratives as in 'the bridge is complete'.

An important aspect of describing a step to the user is selecting an appropriate means of referring to the required objects, which is performed by the output planner and depends on the information in the object inventory. The initial object inventory for the sample interaction is shown in Table 1. When the system generates the SYSTEM[2] utterance in the second (top-down) extract, the back and the front do not yet exist, so they are referred to indefinitely. However, there is one red bolt on the user's table, so a definite is used, while there are 3 relevant slats, so again an indefinite is appropriate. The robot has selected a cube to pick up (if more than one available object matches the description the choice is random), so in this case a demonstrative is used.

4.4 Responding to user actions

Currently the user may respond in a restricted number ways to a system utterance; the range will be extended in the near future, but for now we allow verbal acknowledgements of various kinds, indications of misunderstandings, and yes-no answers. The sample dialogues in Figure 6 contain examples of acknowledgements which are interpreted in different ways. Following SYSTEM[1], 'okay' is interpreted mainly as indicating understanding, while following SYSTEM[2] 'okay' is assumed to indicate that the user has performed the actions described. Depending on the confirmation strategy used by the system, such interpretations might be queried explicitly; the balance between verbal confirmation and the current optimistic grounding approach is another area for experimentation.

The user can indicate that something is misunderstood, in which case previous output is typically repeated. The user may also be asked yes-no questions, which are again interpreted differently depending on the dialogue context. The most obvious example is in cases where the DM reaches a plan step whose result is already listed in the user knowledge set. In this situation the system has the option of asking the user whether or not they remember how to build the object in question.

4.5 Updating the state

Once an assembly step has been completed, the state of the task planner must be updated. As noted above, the DM informs the rest of the system that the step has been executed. This message includes the IDs of the objects that were used, and in response the task planner performs two actions: it updates the set of world objects in the inventory, and it marks the step as completed in its internal AND/OR graph.

Completing an assembly step has two effects on the object inventory. First, all of the components that were involved in the assembly are no longer available for use, so their location is adjusted to indicate that they are part of a larger component. Second, a new object is introduced into the world corresponding to the sub-assembly that was created by the completed step. The updated object inventory after USER[2] in Figure 6 is shown in Table 2, with objects changed by the action indicated by italics.

Table 2: Updated object inventory

ID	Type	Properties	Location
1	*Bolt*	*Color=Red*	*Assembled(11)*
2	Bolt	Color=Orange	Table(Robot)
3	Bolt	Color=Blue	Table(Robot)
4	Cube	Color=Yellow	Table(Robot)
5	*Cube*	*Color=Green*	*Assembled(11)*
6	Cube	Color=Green	Table(User)
7	Nut	Color=Orange	Table(Robot)
8	*Slat*	*Size=3-hole*	*Assembled(11)*
9	Slat	Size=3-hole	Table(User)
10	Slat	Size=5-hole	Table(User)
11	*Comp(front)*	*Parts=(1,5,8)*	*UserHand*

Completing a step also affects the information state. In this case, the user has just built a component called a 'front', so we can update the model of the user's knowledge to indicate that this is likely to be a 'known' object. If a subsequent assembly task also requires a 'front', we can ask the user to build it without needing to explain it in detail, or we can ask the user if they remember the procedure.

5 Discussion

We have described the issues involved in representing assembly plans for use in a task-based dialogue system and shown how we use AND/OR graphs to represent Baufix assembly plans within the JAST human-robot dialogue system. We have then shown how the dialogue manager uses information from the task planner and the object inventory to describe the task plan and the required steps, to respond to actions and requests of the user, and to update the system state following assembly operations.

The dialogue manager has two distinct strategies available for describing a plan. With the top-down strategy, the structure of the plan is described before it is executed; with the depth-first strategy, the dialogue manager proceeds directly to concrete assembly operations and names sub-components only after they are completed. The current JAST system will shortly undergo a user evaluation in which naïve users interact with the system in the current robot-as-instructor scenario. Among other questions, this evaluation will compare these two strategies using measures such as user satisfaction and enjoyment and the success and efficiency of the assembly task.

The system is still under development, and several enhancements are planned for the next version. First, we aim to extend the system to support interactions in which both the robot and the user know the assembly plan. In such scenarios, it is likely that there would be much less verbal interaction between the participants. To support this, we will integrate components from another system (Erlhagen et al., 2007) that addresses a similar human-robot joint assembly task, but that uses dynamic neural fields to infer the user's goals from their non-verbal behaviour and to select complementary actions.

We would like to to move beyond the current small set of simple assembly plans, which are at the moment stored as hard-coded 'recipes' and loaded on request. It would increase the system's flexibility if an AND/OR graph could be created automatically or semi-automatically from a symbolic description of the assembled object; this would also enable the system to learn assembly plans interactively in cooperation with the user. More complex plans could also require different interaction strategies and a different, more flexible connection between the task planner and the dialogue manager in which a single assembly sequence is not selected at the start.

6 Acknowledgements

This work was supported by the EU FP6 IST Cognitive Systems Integrated Project 'JAST' (FP6-003747-IP). We thank the Planning/Language Interest Group at the University of Edinburgh and the Londial reviewers for useful feedback.

References

N. Blaylock and J. Allen. 2005. A collaborative problem-solving model of dialogue. In *Proceedings, 6th SIGdial Workshop on Discourse and Dialogue*, pages 200–211.

C. Breazeal, A. Brooks, J. Gray, G. Hoffman, C. Kidd, H. Lee, J. Lieberman, A. Lockerd, and D. Chilongo. 2004. Tutelage and collaboration for humanoid robots. *International Journal of Humanoid Robotics*, 1(2):315–348. doi:10.1142/S0219843604000150.

O. Brock. 1993. *InterPlan—ein interaktives Planungssystem*. Diplomarbeit (Master's thesis), Technical University of Berlin.

C. Callaway, M. Dzikovska, C. Matheson, J. Moore, and C. Zinn. 2006. Using dialogue to learn math in the LeActiveMath project. In *Proceedings, ECAI 2006 Workshop on Language-Enabled Educational Technology*.

W. Erlhagen, A. Mukovskiy, F. Chersi, and E. Bicho. 2007. On the development of intention understanding for joint action tasks. In *Proceedings, 6th IEEE International Conference on Development and Learning*. doi:10.1109/DEVLRN.2007.4354022.

T. W. Fong, I. Nourbakhsh, R. Ambrose, R. Simmons, A. Schultz, and J. Scholtz. 2005. The peer-to-peer human-robot interaction project. In *AIAA Space 2005*.

B. J. Grosz, S. Weinstein, and A. K. Joshi. 1995. Centering: a framework for modeling the local coherence of discourse. *Computational Linguistics*, 21(2). ACL Anthology J95-2003.

B. Jung. 2003. Task-level assembly modeling in virtual environments. In *Proceedings of Computational Science and Its Applications (ICCSA 2003)*.

A. Knoll. 2003. A basic system for multimodal robot instruction. In P. Kühnlein, H. Rieser, and H. Zeevat, editors, *Perspectives on Dialogue in the New Millennium*, volume 114 of *Pragmatics & Beyond New Series*, pages 215–228. John Benjamins.

S. Kopp, B. Jung, N. Lessmann, and I. Wachsmuth. 2003. Max – a multimodal assistant in virtual reality construction. *Künstliche Intelligenz*, 4(03):11–17.

W. Mann and S. Thompson. 1988. Rhetorical structure theory: toward a functional theory of text organization. *Text*, 3:243–281.

L. S. Homem de Mello and A. C. Sanderson. 1990. AND/OR graph representation of assembly plans. *IEEE Transactions on Robotics and Automation*, 6(2):188–199. doi:10.1109/70.54734.

M. Rickert, M. E. Foster, M. Giuliani, T. By, G. Panin, and A. Knoll. 2007. Integrating language, vision and action for human robot dialog systems. In *Proceedings of HCI International 2007*. doi:10.1007/978-3-540-73281-5_108.

G. Rickheit and I. Wachsmuth. 2006. *Situated Communication*. Mouton de Gruyter, Berlin.

G. Sagerer, C. Bauckhage, E. Braun, J. Fritsch, F. Kummert, F. Lömker, and S. Wachsmuth. 2002. Structure and process: Learning of visual models and construction plans for complex objects. In G. D. Hager, H. I. Christensen, H. Bunke, and R. Klein, editors, *Sensor Based Intelligent Robots*, pages 317–344. Springer.

D. Traum and S. Larsson. 2003. The information state approach to dialogue management. In J. C. J. Van Kuppevelt and R. W. Smith, editors, *Current and New Directions in Discourse and Dialogue*, pages 325–353. Kluwer Academic Publishers.