Towards Programming Multimodal Dialogues

N.L. Vergunst, B.R. Steunebrink, M. Dastani, F.P.M. Dignum, J.-J.Ch. Meyer
Department of Information and Computing Sciences, Utrecht University, The Netherlands
E-mail: {nieske,bass,mehdi,dignum,jj}@cs.uu.nl

Abstract

This paper identifies several issues in multimodal dialogues between a companion robot and a human user. Specifically, these issues pertain to the synchronization of multimodal input and output, and the handling of expected and unexpected input, including input contradicting over different modalities. Furthermore, a novel way of visually representing multimodal dialogues is presented. Ultimately, this work represents some steps towards the development of a principled and generic method for programming multimodal dialogues.

1. Introduction

An increasingly popular field of robotics is that of sociable robots, which typically have a rich arsenal of sensors and actuators allowing users to interact intuitively and affectively with them. An important application of sociable robots is to make them companions that engage in natural interaction, helping humans with daily organizational issues and chores. However, the number of input and output modalities of a typical companion robot presents challenges to developers. The large amount of information going around in a multimodal interaction is both an advantage and a disadvantage: it is usually more robust than a system that only works with a single modality, but on the other hand, misunderstandings may emerge if information in different modalities does not match. This creates the need for synchronization of input and output.

This synchronization takes place on two levels. One notable problem in a multimodal dialogue system is that causally dependent actions in different modalities rarely happen exactly at the same time [8]. Therefore, input to the different modalities often has to be synchronized (e.g., often visual and auditory perception should not be considered separately, but should be processed synchronously), but this holds no less for the output modalities (e.g., speech, lip sync, and facial expressions should match each other).

Secondly, utterances and actions need to be synchronized at the dialogue level. A dialogue is a *joint activity* [4]: a project that is executed by a number of participants together. Joint activities can be split up into smaller parts, called *joint actions*, which are the coordinations of individual actions by two or more people; e.g., answering your dialogue partner's question, or being silent while attending to his or her utterance. Some form of *coordination* is needed to successfully execute these joint actions and activities.

Moreover, a robot can form expectations about a partner's next actions. But then it must also have ways to perform repairs if these expectations are not met. If robots are to behave in such a manner, their multimodal dialogues first have to be properly modeled and then programmed. However, currently there exists no principled and intuitive method for modeling and programming multimodal dialogues with support for expectations. For example, in the work of Breazeal [2] the multimodal interactions emerge from competing behaviors, but they are not synchronized in a principled way.

In this paper, we will look specifically at multimodal dialogues and the issues that synchronization of output and (unexpected) input present to a robot. We will propose a novel way of visually representing modalities and expectations in a dialogue, as well as showing how dialogues can be repaired when things do not proceed as planned.

2. Visually Representing Multimodal Dialogues

In this section, we present an example to illustrate the problem we are addressing and propose steps towards a solution. In this example, iCat [3], a companion robot designed by Philips, has taken up the task of instructing the user to prepare a particular recipe, for which he needs a cooking pan. To make it easier for iCat to observe whether everything is going according to plan, the cooking pans are red and the frying pans are green. In this dialogue, we have abstracted from several topics of future work, such as visual recognition of facial expressions of the user by iCat, other image recognition (e.g., of specific objects in different colors), and parsing user utterances.

iCat: "Please get a cooking pan."
(user takes a red pan)
iCat: "Very good, that's the correct pan."

On the surface, this short conversation consists of only two utterances by iCat and an action by the user, but during the course of this small dialogue, several unseen things also happen. The process starts by iCat *wanting* the user to get a cooking pan, because this is necessary to complete a certain step in the recipe (e.g., cooking pasta), and completing this step is a requirement for completing the whole recipe. Therefore, iCat informs the user that he is supposed to take the cooking pan. After uttering this sentence, iCat *expects* the user to have heard and understood his instruction and to agree with it (and considers this as a common goal of both dialogue participants), and therefore expects the user to perform the action of getting a cooking pan. After iCat has seen the red pan, it confirms that this is the correct pan.

To illustrate in a principled way how dialogue partners coordinate their actions, both internally and externally, we present a representation scheme for multimodal dialogues in the style of a music score (see Figure 1). We visualize this small dialogue in a way that resembles sheet music in several ways. Both dialogue partners have six *tracks* that represent different modalities of input, output, and reasoning, resembling the idea of different instruments that play together in an orchestra. The set of relevant modalities may be different in other dialogue systems or situations, in which case the tracks of the dialogue score can be adapted, just like different pieces of music that may require different instrument groups in an orchestra. Each dialogue partner may be viewed as a section of instruments inside an orchestra, like strings, wood instruments, or a rhythm section. The instruments inside each section are closely connected, like one's own modalities are, but they do interact with the other instrument groups as dialogue partners do.

Everything that happens in the dialogue is represented in the order of occurrence, from left to right. Events that happen (practically) simultaneously are situated directly above/below each other. The arrows show which events are related and depend on each other, in a way that notes in a melody depend on each other: they have to be played in a certain order. Not all of the arrows in Figure 1 have exactly the same meaning: they represent internal or external events. For example, an arrow from a block in the 'reasoning' track to a block in the 'face' or 'speech' track means a signal to execute a plan (internal event), while an arrow from the 'vision' or 'hearing' track to a block in the 'reasoning' track is an external event (input).

Only a hypothetical conductor would have the complete sheet music of the dialogue. Both dialogue partners only have access to their own half of the dialogue score and can only form hypotheses and *expectations* about each other's actions based on the input they get from their dialogue partner and the output they produce themselves. The grey blocks and arrows in the bottom half of Figure 1 are events iCat cannot know or control. iCat presumes that something like this goes on inside the user, and the only way to check this is to synchronize on the black blocks in the bottom half of the dialogue score (e.g., 'takes red pan'): the output by the user. If this *expected event* happens, the dialogue proceeds as planned. The dialogue score in Figure 1 is a typical example of a dialogue that goes exactly as expected. However, if iCat receives input that does not match its expectations, it must attempt to solve the problem. Most notably this can be done by initiating a subdialogue; e.g., asking the user for confirmation or explanation. The exact way of handling such a problem depends on the situation. We illustrate this tactic by introducing an error in the dialogue: the user takes a frying pan instead of a cooking pan.

iCat: "Please get a cooking pan."
(user takes a green pan)
iCat: "That's the wrong pan, the green pans are frying pans. You need a red pan!"
User: "Sorry, I'll get the red one."
(user takes a red pan)
iCat: "Very good, that's the correct pan."

When iCat sees only a green pan (which, as it knows, is a frying pan), its expectations are not met and it revises its plans to deal with the new situation. Several things could have gone wrong: the user might have misunderstood iCat, the red pan might be in the dishwasher, or the user may just not know what a cooking pan is and therefore mistakenly think that he got a cooking pan. While informing the user of his mistake, iCat keeps watching out for the red pan, so that when the user eventually does get the red pan, iCat knows that the goal in question has been achieved, and it informs the user that the actions he performed were correct.

In general, if something else happens instead of the expected event, iCat has to adapt its behavior and expectations about the rest of the dialogue. This is the difference between our representation scheme and an actual music score: while the latter is static, our dialogue score is flexible and can be adapted on-the-fly during the dialogue to suit unexpected behavior. At several points in a dialogue, most notably the user's actions (the black blocks in the lower part of the dialogue score in Figure 1), there is a factor of uncertainty in how the dialogue will proceed: iCat may receive either its desired (expected) input, some kind of unexpected input, or no input at all (within a certain time span). To make the situation even more complicated, iCat may receive input via more than one modality, in which case all these inputs need to be *synchronized* and an appropriate interpretation of the situation and a further course of action should be chosen.

Table 1 shows how iCat handles expected and unexpected input in general, based on its visual input combined with the verbal response from the user. The table distinguishes between three different kinds of events for both modalities. For each type of expected event, some other types of events are specified that contradict with it (e.g., the

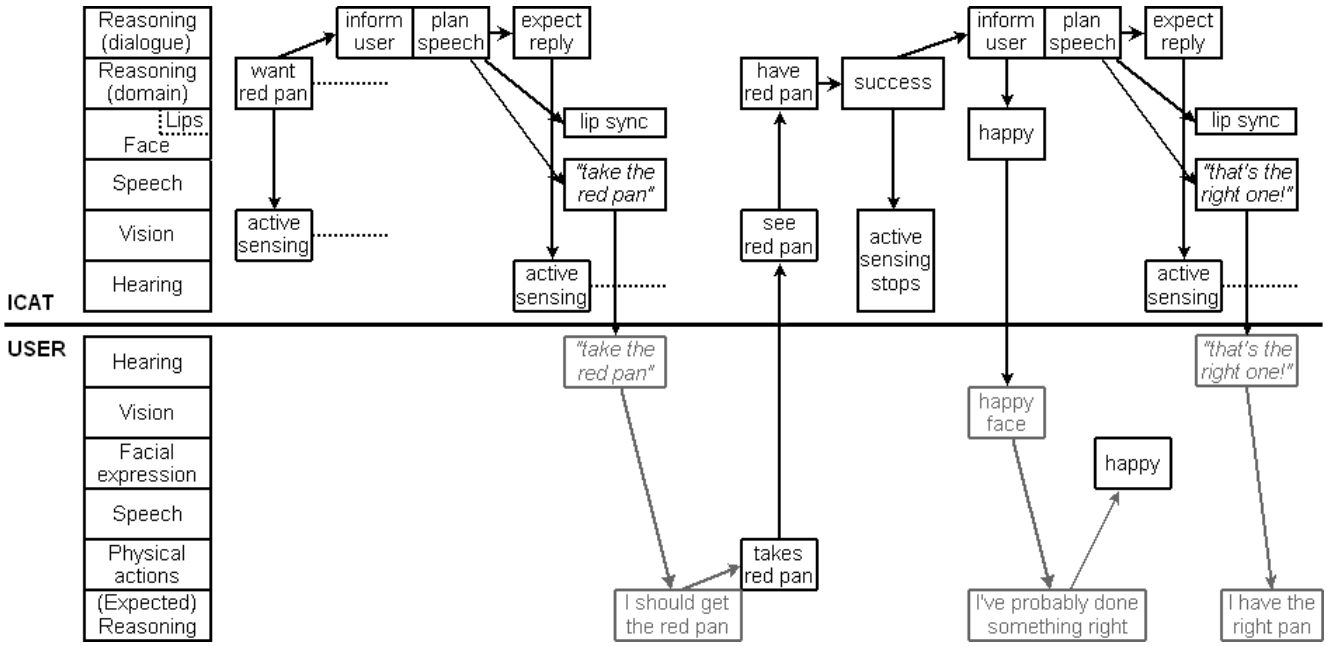

Figure 1. A sample dialogue, in a dialogue score visualization.

event of the user taking a pan of a certain color contradicts with the event of the user taking a pan of a different color). We call these *unexpected events* with respect to a certain expectation. These classes of unexpected events are defined by the programmer and can be as broad or narrow as desired. Note that while it may be difficult to list all these unexpected events in advance, it is usually not a big problem if some are missed. The interactivity of the dialogue gives both the user and iCat the opportunity to address mistakes that are made in the interpretation of input.

	Expected event	**Unexpected event**	**No event**
Expected reply	Everything is ok (1)	Accidental wrong action (2)	Ask for event (3)
Unexpected reply	Inputs clash (4)	Problem; backup plan (5)	Problem; no backup plan (6)
No reply	Everything is ok (7)	Accidental wrong action (8)	Timeout (9)

Table 1. Handling of (un)expected input.

In our example, iCat can either see the correct pan (the *expected event*), a wrong pan (an *unexpected event*), or no pan at all (*no event*). The verbal response from the user is classified in *expected replies* (confirming answers, like "okay" or "I have the pan") and *unexpected replies* (rejecting answers such as "I don't have such a pan" or "it is in the dishwasher"); if the user says nothing, *no reply* is registered.

If iCat receives expected input in both modalities, we are in situation 1 in Table 1: *Everything is ok*. The dialogue will just proceed as planned. However, in case iCat sees a wrong pan (unexpected event) and the user gives a positive answer (expected reply), we are in situation 2: *Accidental wrong action*. The user may have unknowingly grabbed the wrong pan, and iCat's reaction is to inform the user of this mistake: "That's the wrong pan, the green pans are frying pans. You need a red pan!" If the user gives a positive (expected) answer but iCat does not see a pan (no event), iCat will ask the user whether he has really performed the expected action (situation 3: *Ask for event*): "I didn't see you getting a pan. Are you sure you have one?"

There seems to be no good explanation for the situation where iCat sees the correct pan (expected event) but the user gives a negative answer (unexpected reply). In this situation (4: *Inputs clash*), iCat will ask the user for explanation. If the user gives a negative answer (unexpected reply) and iCat sees the wrong pan (unexpected event), iCat concludes that the user has a problem in getting the right pan (maybe it is in the dishwasher). However, the user chooses to follow a backup plan: he (willingly) takes a different pan (situation 5: *Problem; backup plan*). In this case, iCat gives the user full responsibility for unexpected consequences of his action (e.g., "You have a different pan than the recipe says, but if you think it's okay, you can use this one too."). In case of a negative answer (unexpected reply) and no pan (no event), apparently the plan cannot go ahead as expected and the user does not have a backup plan ready (situation

6: *Problem; no backup plan*). iCat should ask the user if he has any alternatives or if he wants to stop or pause the process (e.g., to wash the needed pan).

If iCat sees the right pan (expected event) but the user says nothing (no reply), the situation (7: *Everything is ok*) is similar to situation 1: there is no reason to suspect something is wrong and the dialogue can proceed as planned. If the user gets a wrong pan (unexpected input) and says nothing (no reply), iCat presumes that the user made a mistake (situation 8: *Accidental wrong action*), and will attempt to repair the situation just like in situation 2 described above. Finally, if iCat sees no pan and hears no utterance from the user (situation 9: *Timeout*), it will wait for a while and then ask the user if he is still participating in the joint activity.

With respect to converting a dialogue score to a program, the different blocks in the score can essentially be viewed as parallel plans that have to be synchronized (where plans are sequences of basic actions). If different tracks contain blocks at overlapping time points in the dialogue score, then there are plans that have to be executed in parallel. Moreover, in most cases these parallel plans have to be synchronized (e.g., facial animation should be timed with the speech synthesis for lip synchronization and word emphasizing animations). This poses a challenge for programming the behavior of the robot. The dialogue score visualization helps by identifying which tracks and blocks there are in a particular scenario, and how the flow of information and control between blocks is organized. There should then be an agent programming language [1] that facilitates the construction of parallel plans and has ways of dealing with synchronization of parallel plans. Furthermore, the language should facilitate handling of expectations as in Table 1. Although several agent programming languages may already allow such features to be programmed, we propose to introduce new syntax and corresponding semantics to make the process easy and intuitive. Because of space limitations, we cannot present the formal semantics of such an extension here, but its workings have been sketched in this section and remain a topic of future work.

3. Conclusions and Future Research

To conclude this paper, we will first compare our approach to some related work. Q [6] is an extension of the Scheme programming language. Q allows for interaction scenarios to be programmed in term of *cues* (interaction-triggering events) and *actions*, but it abstracts from the semantics so that no assumptions about the implementations of autonomous robots have to be made. This abstraction from agent implementation is similar to our approach; in our music score visualization, we also do not directly specify how the agents are implemented. The difference between Q and our approach is that we visualize the flow of a complete dialogue in a way that resembles a music score, while Q does not use such a visual representation, but instead enables scenario writers and interaction designers to define certain situations through the use of *interaction pattern cards* in a Microsoft Excel interface.

IOM/T (Interaction Oriented Model by Textual representation) [5] is an interaction description language with a syntax similar to that of the Java programming language. In its design, care has been taken not to disperse an interaction into multiple agents, but to stay as close as possible to the AUML sequence diagrams to which the implementation of an interaction should correspond. In fact, IOM/T has been shown to be formally equivalent to AUML sequence diagrams [5]. The similarity between our approach and IOM/T is that both use a visual representation of a dialogue. However, after making an AUML sequence diagram in IOM/T, the agents still need to be implemented in a Java-like programming language. We are planning to develop a tool to translate the music scores into programs automatically.

In this paper we have identified and discussed some issues in multimodal dialogues. First, the multimodal output of a robot needs to be synchronized to make sense to its dialogue partner. Secondly, a robot needs to synchronize its plans on expected actions of its partner. Thirdly, in case of an unexpected event, a robot needs to have ways to react to these events and repair the remainder of the dialogue. We have presented a principled approach for visually modeling multimodal dialogues and a scheme for reacting to different combinations of expected, unexpected, and absent events. The visual dialogue score representation can be used to identify issues that arise in multimodal dialogues.

References

[1] Bordini, R.H., Dastani, M., Dix, J., Seghrouchni, A.E.F., *Multi-Agent Programming: Languages, Platforms and Applications*, Springer, 2005.

[2] Breazeal, C.L., *Designing Sociable Robots*, MIT Press, 2002.

[3] Breemen, A.J.N. van, *iCat: Experimenting with Animabotics*, AISB 2005 Creative Robotics Symposium, Hatfield, England, 2005.

[4] Clark, H.H., *Using Language*, Cambridge University Press, 1996.

[5] Doi, T., Tahara, Y., Honiden, S., *IOM/T: An Interaction Description Language for Multi-Agent Systems*, Proc. of AAMAS'05, Utrecht, the Netherlands, 2005.

[6] Ishida, T., *Q: A Scenario Description Language for Interactive Agents*, IEEE Computer, 35(11):42–47, 2002.

[7] Oviatt, S., *Integration and Synchronization of Input Modes during Multimodal Human-Computer Interaction*, Proc. of ACL/EACL'97, Madrid, 1997.

[8] Oviatt, S., *Ten Myths of Multimodal Interaction*, Communications of the ACM, 1999.