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, PO Box 80.089 3508TB Utrecht

The number of input and output modalities of a typical companion robot presents challenges to developers. In an obvious way this is because output to different modalities often has to be synchronized (e.g., speech, lip sync, and facial expressions should match each other). But this holds no less for input modalities. Often visual and auditory perception should not be considered separately, but processed synchronously. A complicating factor is that causally dependent actions in different modalities rarely happen exactly at the same time [5]. Moreover, in multimodal dialogues, a robot can form expectations about a partner's next actions. But then it must also have ways to handle unexpected input. 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. We present a representation scheme for multimodal dialogues, constituting some steps towards a principled method for programming them.

To illustrate what happens beneath the surface of a dialogue, the companion robot iCat [3], has taken up the task of instructing the user to prepare a particular recipe, for which he needs a cooking pan.

iCat: "Please get a cooking pan."

(user takes a red pan)

iCat: "Very good, that's the correct pan."

After uttering this instruction to the user, 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, which it knows is a cooking 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

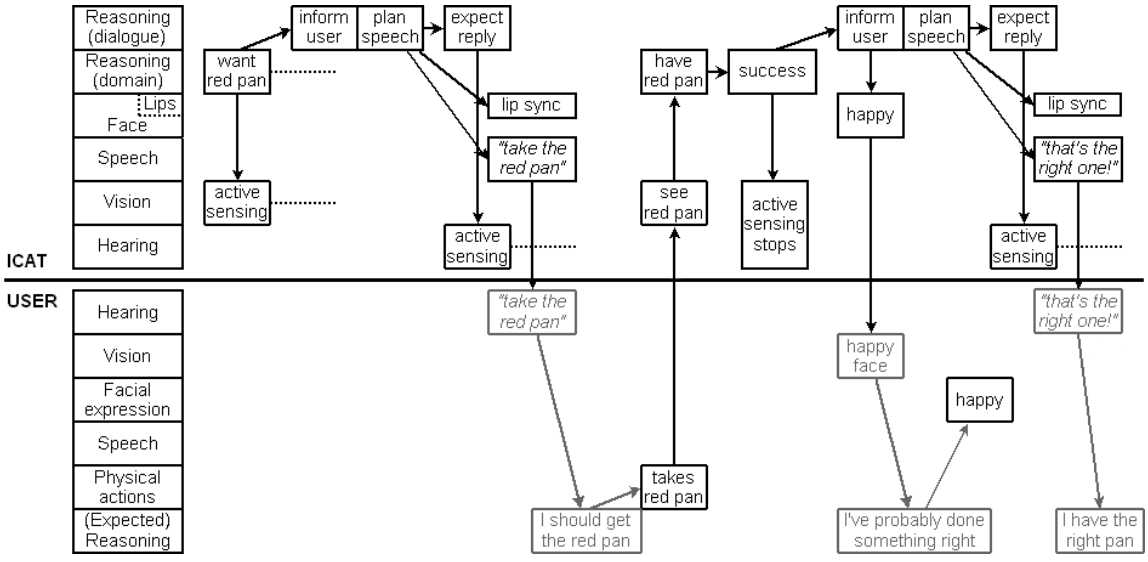

Figure 1: A sample dialogue, in a dialogue score visualization.

Figure 1). We visualize this small dialogue in a way that resembles sheet music in several ways. A dialogue, being a *joint activity* of two participants [4], is played like a duet. 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. 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 *expectations* about each other's internal actions based on the input they get from their dialogue partner and the output they produce. 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 will attempt to solve the problem, adapting 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 real-time 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). As a complicating factor, 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.

	Expected event	**Unexpected event**	**No event**
Expected reply	Everything is ok	Accidental wrong action	Ask for event
Unexpected reply	Inputs clash	Problem; backup plan	Problem; no backup plan
No reply	Everything is ok	Accidental wrong action	Timeout

This table shows how iCat handles expected and unexpected input in general, based on its visual input combined with the verbal response from the user. For each type of expected event, the programmer specifies some other types of events that contradict with it (e.g., the event of the user taking a pan of a certain color contradicts with the user taking a pan of a different color). We call these *unexpected events* w.r.t. a certain expectation. The table distinguishes between three different kinds of events for both these modalities. For visual input, 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 "that pan is in the dishwasher"); if the user says nothing, *no reply* is registered.

For example, if iCat receives expected input in both modalities, we are in situation *Everything is ok* as represented in the top left of the table. The dialogue will just proceed as planned. However, in case iCat sees a wrong pan (unexpected event) and the user gives a negative answer (unexpected reply), we are in situation *Problem; backup plan*, where the user is assumed to have a backup plan and wants to use this 'wrong' pan.

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. Blocks at overlapping time points in the dialogue score are plans that have to be executed in parallel. Moreover, in most cases these parallel plans have to be synchronized (e.g., speech, emphasizing facial animation, and lip sync should be timed). 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 provides ways of dealing with synchronization of parallel plans. Furthermore, the language should facilitate handling of expectations as in the table. In future work, we plan to use the visual dialogue score representation as input to a tool for the automatic generation of program code.

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'05, Hatfield, England, 2005.

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

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