Turn-taking with a hidden agenda

Robin Cooper
Centre for Linguistic Theory and Studies in Probability (CLASP)
Department of Philosophy, Linguistics and Theory of Science
University of Gothenburg
cooper@ling.gu.se

Abstract

We propose a simple model of turn-taking in an information state based approach to dialogue using TTR (Type Theory with Records). The information state (dialogue gameboard) contains an agenda formulated as a list of speech event types that the dialogue participant plans to realize. A novel aspect of the proposal is that the agenda also includes types of events that intuitively should be carried out by an interlocutor. We argue that all dialogue events should be regarded as events jointly carried out by the dialogue participants and that this yields a simple formal method for representing turn-taking in a formal treatment of dialogue.

1 Perception and types

In the literature on TTR (Type Theory with Records), see Cooper and Ginzburg (2015) for a recent introduction, a connection is made between the notion of judgement in type theory (judging that an object or event is of a certain type) and perception, that is, perception involves classifying something as being of a certain type. We will describe this in this section. As we interact with our environment we not only perceive objects but also create new objects of certain types. Performing an action is creating an event of a particular type. A plan is a list of types which we hope to realize in this way. Thus we obtain a simple theory of action based on type theoretic ideas, which we will describe in Section 2. In Section 3 we will consider how coordinated action can be modelled in terms of games in this framework. We will see in Section 4 that this type theoretical view of action leads naturally to a notion of joint action and that this is important in order to obtain a theory of coordinated action. Finally, in Section 5, we will apply this view of action to turn taking in dialogue.

TTR is a type theory which takes many ideas from Martin-Löf type theory (Martin-Löf, 1984; Nordström et al., 1990). This kind of type theory differs from the version of the simple theory of types that Montague used (Montague, 1973; Montague, 1974) in that it allows for a rich collection of types including types like *Dog* and, following a suggestion by Ranta (1994), types of situations like *A_boy_hug_a_dog* in addition to the kind of types corresponding to basic ontological categories (for example, in Montague's case, types like *Entity* and *Truth_value*) and all types of functions based on the basic types which are introduced in simple type theory. Central to this kind of type theory is the notion of a judgement that an object a is of a type T, in symbols, $a : T$. We will sometimes refer to a as a *witness* for the type T. In the literature on TTR this notion of judgement is connected to a theory of perception. An act of perception involves making such a type judgement. When we perceive something we perceive it as being of a certain type. That is, perceiving an object a as a dog involves making the type judgement a : *Dog*. Similarly perceiving a situation, e, as one in which a boy hugs a dog involves making the type judgement e : *A_boy_hug_a_dog*. Agents are thought of as having a collection of types available as a resource which they can employ in, among other things, acts of perception. The types available to an agent are in part limited by their perceptual apparatus.

2 Action and types

A judgement can be thought of as an action which an agent carries out, for example, when an object is presented to its perceptual apparatus. Cooper (2014; Cooper (in prep) calls it a kind of *type act* (meant as a parallel to speech act) and presents a simple theory of action based on types. Basically, there are three things that you can do with types: (i) judge an object to be of a type (ii) wonder whether an object is of a type (iii) create a new object of a type. The third of these is important for this paper. Since we have types of situations (including events) we can regard actions as involving the creation of a situation of a certain type. Consider a particular boy, b and a particular dog, d. The type of situation in which b hugs d can be represented in TTR as a *ptype* (a type constructed from a predicate and appropriate arguments), 'hug(b,d)'. It is no accident that the notation for this situation type is the same as that for a logical formula corresponding to a proposition. In this kind of type theory, types can serve as propositions (the "propositions as types" slogan[1]). When considered as propositions, they are true if there is something of the type and false if there is nothing of the type. Thus if b creates a situation of the type 'hug(b,d)', then b has guaranteed that the type is "true".

3 Coordinated action and games

Let us consider a slightly more complicated kind of situation. Suppose we have a human and a dog playing the game of fetch, where the human picks up a stick and throws it, the dog runs after the stick and brings it back to the human. This involves a string of events[2] which could be regarded as witnesses for ptypes in TTR. If $T_1, \ldots, T_n$ are types, then $T_1{}^\frown \ldots {}^\frown T_n$ is a type whose witnesses are strings $a_1 \ldots a_n$ such that $a_i : T_i$ for $1 \leq i \leq n$. Thus a game of fetch between a human, a, and a dog, b, involving a stick, c could be characterized as having the type:

$$\text{pick\_up}(a,c)^\frown\text{attract\_attention}(a,b)^\frown\text{throw}(a,c)^\frown \text{run\_after}(b,c)^\frown\text{pick\_up}(b,c)^\frown\text{return}(b,c,a)$$

For technical reasons that will become apparent below we will use record types containing these ptypes instead of the simple ptypes:

$$\Big[\text{e:pick\_up}(a,c)\Big]^\frown\Big[\text{e:attract\_attention}(a,b)\Big]^\frown \Big[\text{e:throw}(a,c)\Big]^\frown\Big[\text{e:run\_after}(b,c)\Big]^\frown \Big[\text{e:pick\_up}(b,c)\Big]^\frown\Big[\text{e:return}(b,c,a)\Big]$$

This gives us a label 'e' which we can use as a pointer to pick out the individual subevents. A record, $\Big[\text{e=s}\Big]$, would be of type $\Big[\text{e:pick\_up}(a,c)\Big]$ just in case s:pick_up(a,c). We can think of a record as modelling a situation with one or more facts which hold in it. Witnesses for record types may contain more facts than those required by the type. Thus $\begin{bmatrix}\text{e} = s \\ \text{e}' = s'\end{bmatrix}$ would also be a witness of this record type just in case the object, s, in the field labelled by 'e' is of the appropriate type. Fields with labels not mentioned in the type are ignored.

One aspect of coordination between the human and the dog is that they both realize that the game they are playing has this type. For each type in the string an event of that type has to be created and this has to be carried out in the appropriate order, of course. One way to do this is to think of the rules of the game as a collection of update functions addressing an agenda in the agents' information state. An agenda is a list of types (that is, it is of type '[*Type*]') which the agent plans to realize in order. An update function can come in one of two forms. The first form will map an information state of a given type to a new type which can then be used to compute a type for the new information state. The second form will map an information state of a given type and an event of a given type to a new type which can be used to compute a type for the new information state. The type of information state we are using here is $\Big[\text{agenda:}[\textit{RecType}]\Big]$, that is the type of records which have a field labelled 'agenda' which contains a list of record types. (*RecType* is the type of record types and [*RecType*] is the type of lists of record types.) We can restrict the type of information states to be one where the agenda is required to be some specific list, L, by using a manifest field: $\Big[\text{agenda=}L\text{:}[\textit{RecType}]\Big]$, the type of information states whose 'agenda'-field contains the list, L.

The two forms of update function are illustrated with respect to the fetch game below.

$$\lambda r{:}\Big[\text{agenda=[]:}[\textit{RecType}]\Big]\ .\ \Big[\text{agenda=[}\Big[\text{e:pick\_up}(a,c)\Big]\text{]:}[\textit{RecType}]\Big]$$

This function maps a state with an empty agenda to the type of states where the agenda contains a sole member the type of situation where a picks up c.

$$\lambda r{:}\Big[\text{agenda=[}\Big[\text{e:pick\_up}(a,c)\Big]\text{]:}[\textit{RecType}]\Big]\ .$$

[1] See Wadler (2015) for an account of the origins of this slogan from the perspective of computer science and Ranta (1994) for a discussion of its relevance for linguistic semantics

[2] The idea of events as strings come from work by Tim Fernando, most recently presented in Fernando (2015).

λe: $\begin{bmatrix}\text{e:pick\_up}(a,c)\end{bmatrix}$.

$\begin{bmatrix}\text{agenda=[}\begin{bmatrix}\text{e:attract\_attention}(a,b)\end{bmatrix}\text{]:[}\textit{RecType}\text{]}\end{bmatrix}$

This function maps a state with the event type "a picks up c" on the agenda and an event where a picks up c to the type of state which has the type "a attracts b's attention" on the agenda. Such functions can be used by an agent to predict what type of information state could be licensed on the basis of the agent's current information state and, in the case of the second function, also an external event of a given type. The idea is that, if f is such a function of type $T_i \rightarrow$ *RecType* (or $T_i \rightarrow T_e \rightarrow$ *RecType*) and $r : T'_i$ is the current information state where T'_i is a subtype of T_i (and also $e : T'_e$, where T'_e is a subtype of T_e), the type of the next information state is licensed to be $T'_i \boxed{\dot\wedge} f(r)$ (or $T'_i \boxed{\dot\wedge} f(r)(e)$). '$\boxed{\dot\wedge}$' is the operation of *asymmetric merge* (Cooper and Ginzburg, 2015; Cooper, in prep). Basically if one of T_1, T_2 is not a record type then $T_1 \boxed{\dot\wedge} T_2 = T_2$. If T_1, T_2 are both record types, then for labels they do not have in common, $T_1 \boxed{\dot\wedge} T_2$ will contain both the fields from T_1 and T_2. For labels, ℓ, they do have in common, $T_1 \boxed{\dot\wedge} T_2$ will contain a field labelled ℓ with the asymmetric merge of the two types in that field in T_1 and T_2. Asymmetric merge corresponds to the notion of priority unification in the feature-based grammar literature (Shieber, 1986). For example, the asymmetric merge of

$\begin{bmatrix}\text{agenda=[}\begin{bmatrix}\text{e:pick\_up}(a,c)\end{bmatrix}\text{]:[}\textit{RecType}\text{]}\\ \text{other-info:}T\end{bmatrix}$

with

$\begin{bmatrix}\text{agenda=[}\begin{bmatrix}\text{e:attract\_attention}(a,b)\end{bmatrix}\text{]:[}\textit{RecType}\text{]}\end{bmatrix}$

is

$\begin{bmatrix}\text{agenda=[}\begin{bmatrix}\text{e:attract\_attention}(a,b)\end{bmatrix}\text{]:[}\textit{RecType}\text{]}\\ \text{other-info:}T\end{bmatrix}$

An important word in the characterization of update above is *licensed*. Actions are licensed by previous events of the appropriate type as specified by the game. There is, of course, no necessary inference that such an action will occur or even that the type will appear on anybody's agenda. We can at any point decide to stop playing the game. What we can infer is that if we stop in the middle we will not have completed the game and that certain actions are necessary if we are to create an instance of the game type we have in mind. In this way the kind of inferencing that is involved here is enthymematic in the sense of Breitholtz (2014a), Breitholtz (2014b).

4 Joint action to achieve coordination

In Section 3 we have said something about what it might mean for agents to be coordinated on the type of the game they are playing and how they might update their agendas on the basis of previous events considered as events in a particular instance of the game. But we have said nothing about which event types go on which agent's agenda. At first blush it seems there is a clear division of duties between the human and the dog in the game of fetch. The human has to pick up the stick, attract the dog's attention and throw the stick. The dog has to run after it and bring it back to the human. Therefore it might appear that the first three types should, at the appropriate point in the game, appear on the human's agenda and the other two types, again at an appropriate point in the game, appear on the dog's agenda.

But let us think about this a little more carefully before we develop a formal treatment which involves the different types arriving on the appropriate agenda. Suppose the human picks up the stick and tries to realize the event type of attracting the dog's attention. But the dog is facing the other way gnawing on a bone. The human perhaps calls to the dog but gets no response. Perhaps the human walks around the dog so that she is in the dog's line of sight. The dog turns around taking the bone and faces away from the human. The game cannot continue. The dog has to make a contribution to the realization of the type 'attract_attention(a,b)', look at the stick, and look excited, bark or jump up and down or something. The agent who realizes the type is not just the intuitive "first argument" to the predicate. The dog has to give some kind of feed-back that it is up for the game.

Consider another scenario, a little further on in the game. The stick has been thrown and the dog has run after it and has it in its mouth but then discovers that the human has disappeared. How can the dog realize the type 'return(b,c,a)' if a has wandered off somewhere and is nowhere to be seen? No, the human has to contribute to the realization of this type by at least staying close enough to the dog and in the dog's line of sight when it turns round, quite possibly also by encouraging the dog and looking like she expects the stick to be brought back to her.

These actions are joint actions in the sense of Clark (1996), even if one of the agents is active and the other is fairly passive. Realizing the situation types in a game for two agents is not something that you can do on your own. The technical conclusion I would draw from this is that the types associated with the game are entered onto both agents' agendas as the appropriate juncture in the game as specified by the update functions. Then even if there are types where you don't have to make any kind of active contribution to realizing the type at least there will be a mechanism for causing you to wait until the type on the agenda has been realized before moving on and updating the agenda with a new type. This is known as *turn taking*.

5 Joint action and turn taking in dialogue

I have dwelt at length on the non-linguistic example of the game of fetch because I believe that the basic strategies of coordination, including turn-taking, in dialogue are really the same strategies needed by collaborating agents even without language. The event types involved are very different, involving types of speech events which on an evolutionary scale are extremely specialized and even arcane, but I would like to suggest that the basic turn-taking mechanism which enables coordination in speech is built on the kind of cognitive abilities and strategies necessary for coordinating agents independent of whether they have language or not. This is one reason that it seems important to embed a formal theory of language in a general formal theory of action.

In the literature involving gameboards of the kind Ginzburg has proposed (Ginzburg, 2012) there has not be a great deal of emphasis on getting turn taking to work out. For those of us working with agendas in this kind of framework following (Larsson, 2002), there has been the general assumption that what goes on the agenda are types of events in which the agent is the main actor. Thus, for example, if agent A asks a question of agent B, then the type of the question event first goes on A's agenda and this licenses A to realize an event of this type, that is, ask the question. B, on hearing A's utterance of the question, plans to answer the question, that is, puts a type on B's agenda which is the type of an answer to the question. (This assumes that A and B are playing a straightforward question-answer game rather than something more complicated like a clarification or rejection of the question.) At the point at which B is in this state and utters an answer, A's agenda is empty. There is nothing in such a formal account which represents the fact that when you ask a question you are supposed to wait an appropriate amount of time for an answer and be collaborative. That is, in a normal question-answer exchange you are not supposed to ask a question and then walk out of the room or sing at the top of your voice so that you cannot hear the answer. It seems that the kind of coordination that is required here is exactly like that required between the dog and the human when the dog is picking up the stick. Just like the "passive" role that the human has to play in the returning of the stick, a questioner has a role to play in realizing the type where the question is answered, namely by showing that they are ready to receive the answer. Both A and B should have a type of the answering event on the agenda and jointly play their respective roles in realizing the type. Note that it need not be the case that A and B have exactly the same type on the agenda. For example, A will have a type for a situation where B answers the question. It may be that, at least at some point, before actually answering the question, B has a subtype of A's type on the agenda, namely one that in addition specifies a content for the answer. That is, it is A's job to facilitate an answer, whatever it is. It is B's job to give some particular answer to the question. This shows that a notion of A and B being coordinated does not necessarily involve having the *same* types on their respective agendas. But perhaps what counts as coordination is that the respective types stand in the subtype relation and perhaps one could even claim that the type that the main actor of the event has must be a subtype of the type that the "supporting" actor has. If it is the other way around then perhaps the supporting actor was expecting something of the main actor that they didn't do in the end – a sign of miscoordination.

6 Conclusion

We have suggested a simple notion of turn taking as a kind of coordination between information states in agents both in linguistic and non-linguistic games and we have emphasized the importance of embedding a formal theory of language in a formal theory of action. It seems on this view that almost any speech act is a kind of joint action, albeit in many cases with a "leading" actor and a "supporting" actor.

References

Ellen Breitholtz. 2014a. *Enthymemes in Dialogue: A mico-rhetorical approach*. Ph.D. thesis, University of Gothenburg.

Ellen Breitholtz. 2014b. Reasoning with topoi – towards a rhetorical approach to non-monotonicity. In *Proceedings of AISB Symposium on "Questions, discourse and dialogue: 20 years after Making it Explicit"*.

Herbert Clark. 1996. *Using Language*. Cambridge University Press, Cambridge.

Robin Cooper and Jonathan Ginzburg. 2015. Type theory with records for natural language semantics. In Lappin and Fox (Lappin and Fox, 2015), pages 375–407.

Robin Cooper. 2014. How to do things with types. In Valeria de Paiva, Walther Neuper, Pedro Quaresma, Christian Retoré, Lawrence S. Moss, and Jordi Saludes, editors, *Joint Proceedings of the Second Workshop on Natural Language and Computer Science (NLCS 2014) & 1st International Workshop on Natural Language Services for Reasoners (NLSR 2014) July 17-18, 2014 Vienna, Austria*, pages 149–158. Center for Informatics and Systems of the University of Coimbra.

Robin Cooper. in prep. Type theory and language: from perception to linguistic communication. Draft of book chapters available from `https://sites.google.com/site/typetheorywithrecords/drafts`.

Tim Fernando. 2015. The Semantics of Tense and Aspect: A Finite-State Perspective. In Lappin and Fox (Lappin and Fox, 2015).

Jonathan Ginzburg. 2012. *The Interactive Stance: Meaning for Conversation*. Oxford University Press, Oxford.

Shalom Lappin and Chris Fox, editors. 2015. *The Handbook of Contemporary Semantic Theory*. In Lappin and Fox (Lappin and Fox, 2015), second edition.

Staffan Larsson. 2002. *Issue-based Dialogue Management*. Ph.D. thesis, University of Gothenburg.

Per Martin-Löf. 1984. *Intuitionistic Type Theory*. Bibliopolis, Naples.

Richard Montague. 1973. The Proper Treatment of Quantification in Ordinary English. In Jaakko Hintikka, Julius Moravcsik, and Patrick Suppes, editors, *Approaches to Natural Language: Proceedings of the 1970 Stanford Workshop on Grammar and Semantics*, pages 247–270. D. Reidel Publishing Company, Dordrecht.

Richard Montague. 1974. *Formal Philosophy: Selected Papers of Richard Montague*. Yale University Press, New Haven. ed. and with an introduction by Richmond H. Thomason.

Bengt Nordström, Kent Petersson, and Jan M. Smith. 1990. *Programming in Martin-Löf's Type Theory*, volume 7 of *International Series of Monographs on Computer Science*. Clarendon Press, Oxford.

Aarne Ranta. 1994. *Type-Theoretical Grammar*. Clarendon Press, Oxford.

Stuart Shieber. 1986. *An Introduction to Unification-Based Approaches to Grammar*. CSLI Publications, Stanford.

Philip Wadler. 2015. Propositions as Types. *Communications of the ACM*, 58(12):75–84.