Joint Proceedings NLCS'14 & NLSR 2014, July 17-18, 2014, Vienna, Austria

How to do things with types[*]

Robin Cooper

University of Gothenburg

Abstract

We present a theory of type acts based on type theory relating to a general theory of action which can be used as a foundation for a theory of linguistic acts.

Contents

1 Introduction

The title of this paper picks up, of course, on the seminal book *How to do things with words* by J.L. Austin [Aus62], which laid the foundation for speech act theory. In recent work on dialogue such as [Gin12] it is proposed that it is important for linguistic analysis to take seriously a notion of linguistic acts. To formulate this, Ginzburg uses TTR ("Type Theory with Records") as developed in [Coo12]. The idea was also very much a part of the original work on situation semantics [BP83] which has also served as an inspiration for the work on TTR.

TTR was also originally inspired by work on constructive type theory [ML84, NPS90] in which there is a notion of judgement that an object a is of a type T, $a : T$.[1] We shall think of judgements as a kind of type act and propose that there are other type acts in addition to judgements. We shall also follow [Ran94] in including types of events (or more generally, situations) in a type theory suitable for building a theory of action and of linguistic meaning.

[*]The work for this paper was supported in part by VR project 2009-1569, SAICD.

[1]While TTR has borrowed liberally from the many important ideas in constructive type theory, it does not adhere rigidly to the intuitionistic programme of the original and it has features, such as intersection and union types and a general model theoretic approach, which might make some type theorists judge it not to be a type theory at all. We have found it productive, however, to relate the ideas from modern type theory to the classical model theoretic approach adopted in formal semantics stemming from the original work by [Mon74].

This paper attempts to explore what kind of general theory of action such a notion of linguistic acts could be embedded in and tries to develop the beginnings of such a theory using type theory as the starting point. We will here only develop a non-linguistic example, although we will point out relationships to linguistic acts as we go along.

2 Type acts

Imagine a boy and a dog playing a game of fetch. (The boy throws a stick and the dog runs after it and brings it back to the boy.) The boy and the dog have to coordinate and interact in order to create an event of the game of fetch. This involves doing more with types than just making judgements. For example, when the dog observes the situation in which the boy raises the stick, it may not be clear to the dog whether this is part of a fetch-game situation or a stick-beating situation. The dog may be in a situation of entertaining these two types as possibilities prior to making the judgement that the situation is of the fetch type. We will call this act a query as opposed to a judgement. Once the dog has made the judgement[2] what it has observed so far is an initial segment of a fetch type situation it has to make its own contribution in order to realize the fetch type, that is, it has to run after the stick and bring it back. This involves the creation of a situation of a certain type. Thus creation acts are another kind of act related to types. Creating objects of a given type often has a *de se* [see, for example, Per79, Lew79, Nin10, Sch11] aspect. The dog has to know that it itself must run after the stick in order to make this a situation in which it and the boy are playing fetch. There is something akin to what Perry calls an essential indexical here, though, of course, the dog does not have indexical linguistic expressions. It is nevertheless part of the basic competence that an agent needs in order to be able to coordinate its action with the rest of the world that it has a primitive sense of self which is distinct from being able to identify an object which has the same properties as itself. We will follow Lewis in modelling *de se* in terms of functional abstraction over the "self". In our terms this will mean that *de se* type acts involve dependent types.

In standard type theory we have judgements such as $o : T$ "o is of type T" and T *true* "there is something of type T". We want to enhance this notion of judgement by including a reference to the agent A which makes the judgement, giving judgements such as $o :_A T$ "agent A judges that o is of type T" and $:_A T$ "agent A judges that there is some object of type T". We will call the first of these a *specific* judgement and the second a *non-specific* judgement. Such judgements are one of the three kinds of acts represented in (1) that we want to include in our type act theory. The three kinds of

[2]What judgement is made and what queries are entertained we imagine being governed by some kind of Bayesian learning theory. An initial suggestion along these lines is made by [CDLL14a] who base the learning theory on a string of probabilistic Austinian propositions, records consisting of a situation, s, a type, T and a probability, p, corresponding to a judgement by the learning agent that s if of type T with probability p.

(1) *Type Acts*

judgements

specific $o :_A T$ "agent A judges object o to be of type T"

non-specific $:_A T$ "agent A judges that there is some object of type T"

queries

specific $o :_A T?$ "agent A wonders whether object o is of type T"

non-specific $:_A T?$ "agent A wonders whether there is some object of type T"

creations

non-specific $:_A T!$ "agent A creates something of type T"

type acts in (1) underly what are often thought of as core speech acts: assertion, query and command. Note that creations only come in the non-specific variant. You cannot create an object which already exists.

Creations are also limited in that there are certain types which a given agent is not able to realize as the main actor. Consider for example the event type involved in the fetch game of the dog running after the stick. The human cannot be the main creator of such an event since it is the dog who is the actor. The most the human can do is wait until the dog has carried out the action and we will count this as a creation type act. This will become important when we discuss coordination in the fetch-game below and it is also important in accounting for turn-taking in the coordination of dialogue. It is actually important that the human makes this passive contribution to the creation of the event of the dog running after the stick and does not, for example, get the game confused by immediately throwing another stick before the dog has had a chance to retrieve the first stick. There are other cases of event types which require a less passive contribution from an agent other than the main actor. Consider the type of event where the dog returns the stick to the human. The dog is clearly the main actor here but the human has also a role to play in making the event realized. For example, if the human turns her back on the dog and ignores what is happening or runs away the event type will not be realized despite the dog's best efforts. Something similar holds in language. If it is your dialogue partner's turn to make an utterance, you still have to play your part by paying attention and trying to understand. Other event types, such as lifting a piano, involve more equal collaboration between two or more agents, where it is not intuitively clear that any one of the agents is the main actor. So when we say "agent A creates something of type T" perhaps it would be more accurate to phrase this as "agent A contributes to the creation of something of type T" where A's contribution might be as little as not realizing any of the other types involved in the game until T has been realized.

De se type acts involve functions which have the agent in its domain and return a type, that is, they are dependent types which, given the agent, will yield a type. We will say that agents are of type *Ind* ("individual") and that the relevant dependent types, $\mathcal{T}$, are functions of type $(Ind \rightarrow Type)$. We characterize *de se* type acts in a way parallel to (1), as given in (2).

(2) *De Se Type Acts*

judgements

specific $o :_A \mathcal{T}(A)$ "agent A judges object o to be of type $\mathcal{T}(A)$"

non-specific $:_A \mathcal{T}(A)$ "agent A judges that there is some object of type $\mathcal{T}(A)$"

queries

specific $o :_A \mathcal{T}(A)?$ "agent A wonders whether object o is of type $\mathcal{T}(A)$"

non-specific $:_A \mathcal{T}(A)?$ "agent A wonders whether there is some object of type $\mathcal{T}(A)$"

creations

non-specific $:_A \mathcal{T}(A)!$ "agent A creates something of type $\mathcal{T}(A)$"

From the point of view of the type theory *de se* type acts seem more complex than non-*de se* type acts since they involve a dependent rather than a non-dependent type and a functional application of that dependent type to the agent. However, from a cognitive perspective one might expect *de se* type acts to be more basic. Agents which perform type acts using types directly related to themselves are behaving egocentrically and one could regard it as a more advanced level of abstraction to consider types which are independent of the agent. This seems a puzzling way in which our notions of type seem in conflict with out intuitions about cognition.

While these type acts are prelinguistic (we need them to account for the dog's behaviour in the game of fetch) we will try to argue later that they are the basis on which the notion of speech act [Aus62, Sea69] is built. The idea would be that our division of type acts into the three classes judgements, queries and creations underly the core speech acts assertion, questions and imperatives and that other kinds of speech acts are related to one of the three kinds of type acts. Our notion of using types in query acts seems intuitively related to work on inquisitive semantics [GR12] where some propositions (in particular disjunctions) are regarded as inquisitive. However, this will still allow us to make a distinction between questions and assertions in natural language as argued for by [Gin12] and [GCF14].

3 Coordinating interaction in games

Let us now apply these notions to the kind of interaction that has to take place between the human and the dog in a game of fetch. First consider in more detail what is actually involved in playing a game of fetch, that is creating an event of the particular type represented by "game of fetch". Each agent has to keep track in some way of where they are in the game and in particular what needs to happen next. We analyze this by saying that each agent has an information state which we will model as a record. We need to keep track of the progression of types of information state for an agent during the course of the game. We will refer to the types of information states as gameboards. Our notions of *information state* and *gameboard* are taken from [Lar02] and [Gin12] respectively as well as a great deal of related literature on the gameboard or information state approach to dialogue analysis originating from [Gin94]. We have adapted the notions somewhat to our own purposes but we want to claim that the kind of information states that have been proposed in the literature on dialogue are developments from the kind of information states needed to account for non-linguistic coordination. The idea is that as part of the event occurs then the agent's gameboard is updated so that an event of the next type in the string is expected. For now, we will consider gameboards which only place one requirement on information states, namely that there is an agenda which indicates the type of the next move in the game. Thus if the agent is playing fetch and observes an event of the type where the human throws the stick, then the next move in the game will be an event of the type where the dog runs after the stick. If the actor in the next move is the agent herself then the agent will need to create an event of the type of the next move if the game is to progress. If the actor in the next move is the other player in the game, then the agent will need to observe an event and judge it to be of the appropriate type in order for the game to progress. The type of information states, *InfoState*, will be (3a). (In dialogue, we see more complex information states which include additional fields in the record types.) The type of the initial information state, *InitInfoState*, will be one where the agenda is required to be the empty list.

(3) a. $\left[\ \text{agenda}\ :\ [\mathit{RecType}]\ \right]$

b. $\left[\ \text{agenda=[]}\ :\ [\mathit{RecType}]\ \right]$

((3b) is a manifest field as discussed in [Coo12] based on an idea by Thierry Coquand. A manifest field restricts the type in the field to the singleton type whose only member is the object represented after '='.) We can now see the rules of the game corresponding to the type as a set of update functions which indicate for an information state of a given type what type the next information state may belong to if an event of a certain type occurs. These update functions correspond to the transitions in a finite state machine. This is given in (4).

$$
\begin{array}{lll}
(4) & \{ & \lambda r{:}\big[\text{agenda}{=}[]{:}[\mathit{RecType}]\big]\ .\\
& & \quad \big[\text{agenda}{=}[\big[\text{e:pick\_up}(a,c)\big]]{:}[\mathit{RecType}]\big],\\
& & \lambda r{:}\big[\text{agenda}{=}[\big[\text{e:pick\_up}(a,c)\big]]{:}[\mathit{RecType}]\big]\\
& & \quad \lambda e{:}\big[\text{e:pick\_up}(a,c)\big]\ .\\
& & \qquad \big[\text{agenda}{=}[\big[\text{e:attract\_attention}(a,b)\big]]{:}[\mathit{RecType}]\big],\\
& & \lambda r{:}\big[\text{agenda}{=}[\big[\text{e:pick\_up}(a,c)\big]]{:}[\mathit{RecType}]\big]\\
& & \quad \lambda e{:}\big[\text{e:attract\_attention}(a,b)\big]\ .\\
& & \qquad \big[\text{agenda}{=}[\big[\text{e:throw}(a,c)\big]]{:}[\mathit{RecType}]\big],\\
& & \lambda r{:}\big[\text{agenda}{=}[\big[\text{e:throw}(a,c)\big]]{:}[\mathit{RecType}]\big]\\
& & \quad \lambda e{:}\big[\text{e:throw}(a,c)\big]\ .\\
& & \qquad \big[\text{agenda}{=}[\big[\text{e:run\_after}(b,c)\big]]{:}[\mathit{RecType}]\big],\\
& & \lambda r{:}\big[\text{agenda}{=}[\big[\text{e:run\_after}(b,c)\big]]{:}[\mathit{RecType}]\big]\\
& & \quad \lambda e{:}\big[\text{e:run\_after}(b,c)\big]\ .\\
& & \qquad \big[\text{agenda}{=}[\big[\text{e:pick\_up}(b,c)\big]]{:}[\mathit{RecType}]\big],\\
& & \lambda r{:}\big[\text{agenda}{=}[\big[\text{e:pick\_up}(b,c)\big]]{:}[\mathit{RecType}]\big]\\
& & \quad \lambda e{:}\big[\text{e:pick\_up}(b,c)\big]\ .\\
& & \qquad \big[\text{agenda}{=}[\big[\text{e:return}(b,c,a)\big]]{:}[\mathit{RecType}]\big],\\
& & \lambda r{:}\big[\text{agenda}{=}[\big[\text{e:return}(b,c,a)\big]]{:}[\mathit{RecType}]\big]\\
& & \quad \lambda e{:}\big[\text{e:return}(b,c,a)\big]\ .\\
& & \qquad \big[\text{agenda}{=}[]{:}[\mathit{RecType}]\big]\\
& & \hfill \}
\end{array}
$$

Since we are treating an empty agenda as the condition for the input to the initial state in the corresponding automaton and also the output of the final state we automatically get a loop effect from the final state to the initial state so that the game can be repeated indefinitely many times. In order to prevent the loop we would have to distinguish the type corresponding to the initial and final states. Note that the functions in (4) are of the type (5).

(5) $(\big[\text{agenda}{:}[\mathit{RecType}]\big]{\rightarrow}(\mathit{Rec}{\rightarrow}\mathit{RecType}))$

That is, they map an information state containing an agenda (modelled as a record containing an agenda field) and an event (modelled as a record) to a record type. This is true of all except for the function corresponding to the initial state which is of type (6).

(6) $(\left[\text{agenda}:[\textit{RecType}]\right]\to\textit{RecType})$

That is, it maps an information state directly to a record type and does not require an event. We can think of this set as the set of rules which define the game. It is of the type (7).

(7) $\{((\left[\text{agenda}:[\textit{RecType}]\right]\to(\textit{Rec}\to\textit{RecType}))\vee(\left[\text{agenda}:[\textit{RecType}]\right]\to\textit{RecType}))\}$

Let us call the type in (7) *GameRules*. Sets of game rules of this type define the rules for specific participants as in (4). In order to characterize the game in general we need to abstract out the roles of the individual participants in the game. This we will do by defining a function from a record containing individuals appropriate to play the roles in the game thus revising (4) to (8).

(8)
$$
\begin{array}{l}
\lambda r^*{:}\begin{bmatrix} \text{h} & : & \textit{Ind} \\ \text{c}_{\text{human}} & : & \text{human(h)} \\ \text{d} & : & \textit{Ind} \\ \text{c}_{\text{dog}} & : & \text{dog(d)} \\ \text{s} & : & \textit{Ind} \\ \text{c}_{\text{stick}} & : & \text{stick(s)} \end{bmatrix} . \\
\quad\{\ \lambda r{:}\left[\text{agenda=[]}{:}[\textit{RecType}]\right] . \\
\qquad\quad \left[\text{agenda=}[\left[\text{e:pick\_up}(r^*.\text{h},r^*.\text{s})\right]]{:}[\textit{RecType}]\right], \\
\qquad \lambda r{:}\left[\text{agenda=}[\left[\text{e:pick\_up}(r^*.\text{h},r^*.\text{s})\right]]{:}[\textit{RecType}]\right] \\
\qquad\quad \lambda e{:}\left[\text{e:pick\_up}(r^*.\text{h},r^*.\text{s})\right] . \\
\qquad\qquad \left[\text{agenda=}[\left[\text{e:attract\_attention}(r^*.\text{h},r^*.\text{d})\right]]{:}[\textit{RecType}]\right], \\
\qquad \lambda r{:}\left[\text{agenda=}[\left[\text{e:pick\_up}(r^*.\text{h},r^*.\text{s})\right]]{:}[\textit{RecType}]\right] \\
\qquad\quad \lambda e{:}\left[\text{e:attract\_attention}(r^*.\text{h},r^*.\text{d})\right] . \\
\qquad\qquad \left[\text{agenda=}[\left[\text{e:throw}(r^*.\text{h},r^*.\text{s})\right]]{:}[\textit{RecType}]\right], \\
\qquad \lambda r{:}\left[\text{agenda=}[\left[\text{e:throw}(r^*.\text{h},r^*.\text{s})\right]]{:}[\textit{RecType}]\right] \\
\qquad\quad \lambda e{:}\left[\text{e:throw}(r^*.\text{h},r^*.\text{s})\right] . \\
\qquad\qquad \left[\text{agenda=}[\left[\text{e:run\_after}(r^*.\text{d},r^*.\text{s})\right]]{:}[\textit{RecType}]\right], \\
\qquad \lambda r{:}\left[\text{agenda=}[\left[\text{e:run\_after}(r^*.\text{d},r^*.\text{s})\right]]{:}[\textit{RecType}]\right] \\
\qquad\quad \lambda e{:}\left[\text{e:run\_after}(r^*.\text{d},r^*.\text{s})\right] . \\
\qquad\qquad \left[\text{agenda=}[\left[\text{e:pick\_up}(r^*.\text{d},r^*.\text{s})\right]]{:}[\textit{RecType}]\right], \\
\qquad \lambda r{:}\left[\text{agenda=}[\left[\text{e:pick\_up}(r^*.\text{d},r^*.\text{s})\right]]{:}[\textit{RecType}]\right] \\
\qquad\quad \lambda e{:}\left[\text{e:pick\_up}(r^*.\text{d},r^*.\text{s})\right] . \\
\qquad\qquad \left[\text{agenda=}[\left[\text{e:return}(r^*.\text{d},r^*.\text{s},r^*.\text{h})\right]]{:}[\textit{RecType}]\right], \\
\qquad \lambda r{:}\left[\text{agenda=}[\left[\text{e:return}(r^*.\text{d},r^*.\text{s},r^*.\text{h})\right]]{:}[\textit{RecType}]\right] \\
\qquad\quad \lambda e{:}\left[\text{e:return}(r^*.\text{d},r^*.\text{s},r^*.\text{h})\right] . \\
\qquad\qquad \left[\text{agenda=[]}{:}[\textit{RecType}]\right] \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \}
\end{array}
$$

(8) is of type $(\textit{Rec}\to\textit{GameRules})$ which we will call *Game*.

Specifying the rules of the game in terms of update functions in this way will not actually getting anything to happen, though. For that we need type acts of the kind we discussed. We link the update functions to type acts by means of *licensing conditions on type acts*. A basic licensing condition is that an agent can create (or contribute to the creation of) a witness for the first type that occurs on the agenda in its information state. Such a licensing condition is expressed in (9).

(9) If A is an agent, s_i is A's current information state, $s_i :_A \left[\ \text{agenda}{=}T\,|\,R \ :\ [\mathit{RecType}]\ \right]$, then $:_A\ T!$ is licensed.

Update functions of the kind we have discussed are handled by the licensing conditions in (10).

(10) a. If $f : (T_1 \rightarrow (T_2 \rightarrow \mathit{Type}))$ is an update function, A is an agent, s_i is A's current information state, $s_i :_A T_i$, $T_i \sqsubseteq T_1$ (and $s_i : T_1$), then an event $e :_A T_2$ (and $e : T_2$) licenses $s_{i+1} :_A f(s_i)(e)$.

b. If $f : (T_1 \rightarrow \mathit{Type})$ is an update function, A is an agent, s_i is A's current information state, $s_i :_A T_i$, $T_i \sqsubseteq T_1$ (and $s_i : T_1$), $s_{i+1} :_A f(s_i)$ is licensed.

(10a) is for the case where the update function requires an event in order to be triggered and (10b) is for the case where no event is required. There are two variants of licensing conditions which can be considered. One variant is where the licensing conditions rely only on the agent's judgement of information states and events occurring. The other variant is where in addition we require that the information states and events actually are of the types which the agent judges them to be of. (These conditions are represented in parentheses in (10).) In practical terms an agent has to rely on its own judgement, of course, and there is one sense in which any resulting action is licensed even if the agent's judgement was mistaken. There is another stricter sense of license which requires the agent's judgement to be correct. In the real world, though, the only way we have of judging a judgement to be correct is to look at judgements by other agents.

Licensing conditions will regulate the coordination of successfully realized games like fetch. They enable the agents to coordinate their activity when they both have access to the same objects of type *Game* and are both willing to play. The use of the word "license" is important, however. The agents have free will and may choose not to do what is licensed and also may perform acts that are not licensed. We cannot build a theory that will predict exactly what will happen but we can have a theory which tells us what kinds of actions belong to a game. It is up to the agents to decide whether they will play the game or not. At the same time, however, we might regard whatever is licensed at a given point in the game as an obligation. That is, if there is a general obligation to continue a game once you have embarked on it, then whatever type is placed on an agent's agenda as the result of a previous event in the game can be seen as an obligation on the agent to play its part in the creation of an event of that type.

4 Conclusion

We have sketched the beginnings of a theory of action based on ideas from type theory. We believe that a theory of linguistic acts in terms of updating information states as discussed in the literature on dialogue can be seen as a development of such a basic theory and that this makes a connection between the kind of complex coordination involved in dialogue and coordination that is needed between non-linguistic agents who interact with each other and the rest of the physical world.

References

[Aus62] J. Austin. *How to Do Things with Words.* Oxford University Press, 1962. ed. by J. O. Urmson.

[BP83] Jon Barwise and John Perry. *Situations and Attitudes.* Bradford Books. MIT Press, Cambridge, Mass., 1983.

[CDLL14a] Robin Cooper, Simon Dobnik, Shalom Lappin, and Staffan Larsson. A probabilistic rich type theory for semantic interpretation. In Cooper et al. [CDLL14b], pages 72–79.

[CDLL14b] Robin Cooper, Simon Dobnik, Shalom Lappin, and Staffan Larsson, editors. *Proceedings of the EACL 2014 Workshop on Type Theory and Natural Language Semantics (TTNLS).* Association for Computational Linguistics, Gothenburg, Sweden, April 2014.

[Coo12] Robin Cooper. Type theory and semantics in flux. In Ruth Kempson, Nicholas Asher, and Tim Fernando, editors, *Handbook of the Philosophy of Science*, volume 14: Philosophy of Linguistics, pages 271–323. Elsevier BV, 2012. General editors: Dov M. Gabbay, Paul Thagard and John Woods.

[GCF14] Jonathan Ginzburg, Robin Cooper, and Tim Fernando. Propositions, questions, and adjectives: a rich type theoretic approach. In Cooper et al. [CDLL14b], pages 89–96.

[Gin94] Jonathan Ginzburg. An update semantics for dialogue. In Harry Bunt, editor, *Proceedings of the 1st International Workshop on Computational Semantics*, Tilburg University, 1994. ITK Tilburg.

[Gin12] Jonathan Ginzburg. *The Interactive Stance: Meaning for Conversation.* Oxford University Press, Oxford, 2012.

[GR12] Jeroen Groenendijk and Floris Roelofsen. Course notes on inquisitive semantics, nasslli 2012. Available at `https://sites.google.com/site/inquisitivesemantics/documents/NASSLLI-2012-inquisitive-semantics-lecture-notes.pdf`, 2012.

[Lar02] Staffan Larsson. *Issue-based Dialogue Management.* PhD thesis, University of Gothenburg, 2002.

[Lew79] David Lewis. Attitudes de dicto and de se. *Philosophical Review*, 88:513–543, 1979. Reprinted in [Lew83].

[Lew83] David Lewis. *Philosophical Papers, Volume 1.* Oxford University Press, 1983.

[ML84] Per Martin-Löf. *Intuitionistic Type Theory.* Bibliopolis, Naples, 1984.

[Mon74] Richard Montague. *Formal Philosophy: Selected Papers of Richard Montague.* Yale University Press, New Haven, 1974. ed. and with an introduction by Richmond H. Thomason.

[Nin10] Dilip Ninan. *De Se* Attitudes: Ascription and Communication. *Philosophy Compass*, 5(7):551–567, 2010.

[NPS90] Bengt Nordström, Kent Petersson, and Jan M. Smith. *Programming in Martin-Löf's Type Theory*, volume 7 of *International Series of Monographs on Computer Science.* Clarendon Press, Oxford, 1990.

[Per79] John Perry. The problem of the essential indexical. *Noûs*, 13(1):3–21, 1979. Reprinted in [Per93].

[Per93] John Perry. *The Problem of the Essential Indexical and Other Essays.* Oxford University Press, 1993.

[Ran94] Aarne Ranta. *Type-Theoretical Grammar.* Clarendon Press, Oxford, 1994.

[Sch11] Philippe Schlenker. Indexicality and *De Se* reports. In Claudia Maienborn, Klaus von Heusinger, and Paul Portner, editors, *Semantics: an international handbook of natural language meaning*, pages 1561–1604. de Gruyter, 2011.

[Sea69] John R. Searle. *Speech Acts: an Essay in the Philosophy of Language.* Cambridge University Press, 1969.