Nothing to add

Rick Nouwen

1 Context

Jan van Eijck taught me to be precise. As with models in any discipline, semantic analyses only make sense if we have a detailed understanding of how they work and what their predictions are. Jan often insisted on the value of implementation for this purpose: it allows the exploration and rigorous testing of an analysis, and it creates a fuller appreciation of the complexity of a proposed system. He was ultimately unsuccesful in convincing me that my dissertation should contain a Haskell implementation of the system I was proposing, but we did collaborate on a joint paper in which we presented a semantic analysis of plural discourse anaphora alongside Haskell code implementing that analysis (van Eijck and Nouwen 2002). In this brief note, I want to reconsider one of the design choices we made in that paper.

No doubt the most important line of code in our analysis is the line that determines the view taken on what exactly a context is.

```
type Context = ...
```

It is clear what a context is *for*, that it has a double function in interpretation. On the one hand it provides the necessary background that makes intepretation possible. On the other hand, it is being manipulated by interpretation: by interpreting natural language, we are *changing* the context. To determine what a context *is*, we then need to ascertain what it is that interpretation potentially depends and impacts on.

Doing this, a dual view on context emerges. When you read or listen, you don't just gain potential new beliefs, you also create a memory of the topics and things that are being discussed. In other words, by interpreting natural language we keep track of (i) the topics of conversation and (ii) the information that is being shared on those topics. This way of looking at *interpretation* is the central philosophy behind *dynamic semantics* (Kamp 1981,Groenendijk and Stokhof (1991),van Eijck (2001),Nouwen et al. (2016)), a family of theories of how ordinary semantics can be extended to take into account the basic dynamics of information processing.

In line with this view, Jan van Eijck's Incremental Dynamics (van Eijck 2001) takes a *context* to be a stack of entities (the list of entities that are discussed, if you want)

and interpretation is a function from such stacks to sets of such stacks (those sets that comply with the given information). Different positions in a stack correspond to different *discourse referents*, the entities under discussion. Introducing a new referent in the discourse amounts to simply adding an entity to the context.[1]

$$[\![\exists]\!](c) = \{c^\frown d \mid d \in D\}$$

There are no ordinary variables, only indices, corresponding to locations in the context: $c[n]$ returns the entity stored on the nth position in c. Predication tests are defined as follows:[2]

$$[\![P(n)]\!](c) = \begin{cases} \{c\} & c[n] \in I(P) \\ \emptyset & otherwise \end{cases}$$

Dynamic actions are combined using $[\![\varphi;\psi]\!](c) = \{[\![\psi]\!]c' | c' \in [\![\varphi]\!](c)\}$. And, so, the (in)famous *A man came in. He sighed.* gets the following analysis in Incremental Dynamics, where i is the length of the input context.

$$\exists; man(i); sighed(i)$$

One major advantage of this view on context is that it is absolutely explicit about what the anaphoric potential of the context is: the context *is* the set of accessible discourse referents. As we will see next, this has an interesting consequence once we add quantifiers and plurals to the system.

2 Context and Quantification

In van Eijck and Nouwen (2002), generalised quantifiers work quite similarly to $\exists$ in that they push entities to the stack. This is different from the traditional take on quantifiers in the dynamic semantic literature, most notably different from Kamp (1981), Kamp and Reyle (1993) and Groenendijk and Stokhof (1991). In line with van den Berg (1996), we took quantifiers to simply push the so-called reference set, $A \cap B$ for a statement $Q(A)(B)$, to the context. For instance, *Most students passed the test*, if true, will push the set of all students that passed the test to the input context. The semantic mechanism behind this need not concern us here. Here, I want to focus on the question what the addition of plural discourse referents means for our notion of context.

Since van den Berg (1996), it is standardly assumed that plural referents are not just stored in the context as complex individuals, but that they are rather *distributed* over that context (see also: Nouwen 2007, Brasoveanu (2008)). This is because quantifiers do not just add potentially plural referents to the context, they also store dependencies between referents. For instance,

[1] For those familiar with the literature: what in Dynamic Predicate Logic amounts to a *random reset* is a *random push* in Incremental Dynamics.

[2] Here and in what follows I am ignoring undefinedness cases when n exceeds the length of c.

(1) Every farmer owns exactly one donkey. And most of them beat it.

Here, the first sentence introduces the set of (all) donkey-beating farmers, and the set of all donkeys owned by a farmer. The second sentence shows that we do not just have access to these pluralities, we also have access to which farmer goes with which donkey. This is because the second sentence can only be interpreted as saying that for most farmers, the farmer in question beats the donkey *he or she* owns. If the context is a stack just containing a set of farmers and a set of donkeys, then we have not kept track of this dependency. Instead, we need to say that the context is a set of stacks, each containing one farmer and one donkey:

$$\{f_1d_1, f_2d_2, f_3d_3, \ldots\}$$

The set of entities occurring on the first position in these stack is the relevant set of farmers. The set of entities in second position is the set of donkeys. The dependency is captured by the way these sets are distributed over the set of stacks.

Using my rather rusty Haskell skills, here is what this setup could look like:

```
import Data.List

data Entity = A | B | C | D | E | F | G | H | I | J | K | L | M
         deriving (Eq,Bounded,Ord,Enum,Show)

type StringEnt = [Entity]     -- strings of entities
type Context = [StringEnt]    -- set of strings of entities
type Coll = [Entity]          -- a collection is a set of entities

lookupIdx :: StringEnt -> Int -> Entity
lookupIdx [] i = error "undefined context element"
lookupIdx (x:xs) 0 = x
lookupIdx (x:xs) i = lookupIdx xs (i-1)

lookupIdxCtxt :: Context -> Int -> Coll
lookupIdxCtxt [] i = []
lookupIdxCtxt (x:xs) i = [lookupIdx x i] ++ lookupIdxCtxt xs i

distrib :: Context -> Entity -> Context
distrib [] e = []
distrib (x:xs) e = [x ++ [e]] ++ distrib xs e

push :: Context -> Coll -> Context
push x [] = x
push x [e] = distrib x e
push x (e:es) = distrib x e ++ push x es
```

```
project :: Context -> Int -> Coll
project x i = nub (lookupIdxCtxt x i)

cardi :: Context -> Int -> Int
cardi x i = length ( project x i )
```

To illustrate how this works, here is an example of some operations on an input context $\{AB, AC\}$. This is a context with two discourse referents, one of which is simply A and the other is the plural individual with B and C as its parts (usually written as $B \sqcup C$). The first line shows how this context was created.

```
*Main> push (push [[]] [A]) [B,C]
[[A,B],[A,C]]
*Main> push [[A,B],[A,C]] [D,E,F]
[[A,B,D],[A,C,D],[A,B,E],[A,C,E],[A,B,F],[A,C,F]]
*Main> project (push [[A,B],[A,C]] [D,E,F]) 0
[A]
*Main> project (push [[A,B],[A,C]] [D,E,F]) 1
[B,C]
*Main> project (push [[A,B],[A,C]] [D,E,F]) 2
[D,E,F]
*Main> cardi (push [[A,B],[A,C]] [D,E,F]) 1
2
```

As setup like this will now allow us to provide a general recipe for the dynamic semantics of quantifiers. In general, quantifiers are operators that take two sets and perform a cardinality test on these two sets. If successful, the intersection of the two sets, the reference set, will then be pushed to the stack. Here I give a schematic interpretation of *most* as an example.[3]

$$[\![\text{most}]\!] = \lambda X.\lambda Y.\lambda c.\begin{cases} \texttt{push c } X \cap Y & |X \cap Y| > |X \setminus Y| \\ \emptyset & \text{otherwise} \end{cases}$$

3 Nothing to add

An issue arises as soon as downward entailing quantifiers come into the picture. A quantifier is downward entailing if for any A, B and B' such that $B \subseteq B'$ it is the case that $Q(A)(B') \Rightarrow Q(A)(B)$. The issue concerns the fact that $Q(A)(\emptyset)$ is true for any downward entailing Q and any A. Van Eijck and Nouwen remark:

[3] This is a simplification. The arguments of *most* need to be interpreted in context too, for they may for instance have dynamic effects themselves. In fact, it is this dynamic interpretation of the X and Y argument that ultimately results in the storage of quantificational dependencies. See van den Berg (1996), Nouwen (2003) and Brasoveanu (2008) for discussion.

> "An[…] unsolved problem is the treatment of downward entailing quantifiers. There are essentially two issues. First of all, since we have chosen to make quantifiers dynamic by existentially introducing a new index, we presuppose the existence of a set of individuals satisfying the restrictor and scope of the quantifier relation. Of course, in case the determiner is downward monotone in its right argument, this relation is satisfied even if no such individual exists." (van Eijck and Nouwen, 2002, p.22)

In my dissertation (Nouwen 2003), I offered a way out of this dilemma. In case the reference set is empty, then incrementing is vacuous and consequently there will be nothing to refer to. Pronouns simply won't have the chance to refer back.

This idea already happens to be in place in the short Haskell fragment above (which is different from our 2002 paper):

```
*Main> push [[A,B],[A,D],[A,E]] []
[[A,B],[A,D],[A,E]]
```

That is, pushing the empty set onto the stack simply returns the input stack. No new position is created and, consequently, no new anaphoric potential has been added. This is fully in line with the philosophy of contexts in Incremental Dynamics: contexts are fully explicit about what the anaphoric potential of the context is. Whether or not downward entailing quantifiers introduce a discourse referent depends on the model. Only in models in which the intersection between restrictor and scope is non-empty will there be a new index. Using a pronoun to refer back to the quantifier presupposes that there is such an index and, as such, it presupposes that we are in a model in which that intersection is not empty.

This is an elegant solution to the problem, but one which I am now no longer fully convinced of. The reason is that this solution depends on a standard assumption about collections of entities that is often made in the literature, but which I now think is wrong. It is often assumed that there are two kinds of entities, namely atoms and pluralities, and that the latter are built the former. More precisely, the full set of entities is built by taking the powerset of the set of atomic entities and then removing the empty set. More commonly, the set of entities is actually thought to be only structurally similar to this: plural individuals are not sets but sums and the set of entities is a complete join semi-lattice with the atomic entities as smallest elements (Landman 1989). Explicitly, the assumption is that the structure is a semi-lattice and not a lattice: there is no unique bottom element, that is a part of any other element in the lattice. This is the detail that I now question.

There are (as far as I know) two arguments in the literature that natural language semantics needs to take a bottom element, i.e. an element with cardinality 0, into account. Bylinina and Nouwen (2017) provide a complex argument that the fact that most languages have a word for "0" (for instance, *zero*), entails that there has to be an full-fledged entity with 0 cardinality. The other is Landman (2010), which is more relevant to our discussion of dynamics. Landman notices that some plural definite descriptions appear to refer, even though the referent is empty:

(2) Of the ten students in my class I would say that the students that studied for the test got a good grade (but nobody did).

The parenthetical continuation does not appear contradictory, and so the description *the students that studied for the test* must have referred to the empty individual. Note, especially the contrast with (3), where the singular morphology forces the definite description to refer to an atom, and hence to something that is not empty.

(3) Of the ten students in my class I would say that the student that studied for the test got a good grade (#but nobody did).

On a conceptual level, this observation already goes against my assumption above that empty sets are referentially void. More generally, I believe it is not impossible that Landman's observation extends to pronouns.

(4) Of the ten students in my class I would say that the students that studied for the test got a good grade and that they are likely to do well in the next course. Unfortunately, none of my students studied for the test.

(5) Of the ten students in my class I would say that the student that studied for the test got a good grade and that he is likely to do well in the next course. #Unfortunately, none of my students studied for the test.

I will leave a detailed discussion of these examples for a future occassion. Potentially, they indicate that there can be empty discourse referents. In the terms of Incremental Dynamics this would mean that the operation of *adding nothing to the stack* does not simply return the input stack, but rather returns a stack with an empty value pushed on top of it. In fact, had we chosen not to implement the dependency phenomenon illustrated by (1) but instead taken a simpler view on context as a stack of collections, then we would have arrived at that behaviour straightforwardly. Take the following simpler code:

```
push2 :: Context -> Coll -> Context
push2 x e = x ++ [e]
```

and how it functions:

```
*Main> push2 [[A],[B,C]] []
[[A],[B,C],[]]
```

Here, a context is not a collection of stacks, but a stack of collections. Each position in that stack is a potentially plural and potentially empty entity (represented by a list). In such a setup, pronouns point to positions in the stack and come with a non-emptiness implicature, one that is cancelled in (3).

Of course, the simple `push2` operation above leaves no room to account for examples involving quantificational dependencies like (1). But I have no reason to believe that the absence of an empty element follows from adopting a view of context in which pluralities are distributed over stacks. It just so happened that my crude way of implementing this idea resulted in this behaviour. I leave it as an easy exercise for the reader to come up with a Haskell program that at the same time implements the idea of contexts with distributed pluralities and the idea that pluralities are potentially empty.

References

Brasoveanu, Adrian. 2008. "Donkey Pluralities: Plural Information States Versus Non-Atomic Individuals." *Linguistics and Philosophy* 31 (2): 129–209.

Bylinina, Lisa, and Rick Nouwen. 2017. "The Semantics of 'Zero'."

Groenendijk, J.A.G., and M.J.B. Stokhof. 1991. "Dynamic Predicate Logic." *Linguistics and Philosophy* 14: 39–100.

Kamp, H. 1981. "A Theory of Truth and Semantic Representation." In *Formal Methods in the Study of Language*, edited by J. A. G. Groenendijk, T. M. V. Janssen, and M. J. B. Stokhof. Amsterdam: Mathematical Centre.

Kamp, H., and U. Reyle. 1993. *From Discourse to Logic*. Dordrecht: D. Reidel.

Landman, Fred. 1989. "Groups. Part I, II." *Linguistics and Philosophy* 12: 559–605;723–744.

———. 2010. "Boolean Pragmatics." In *This Is Not a Festschrift*, edited by Jaap van der Does and Catarina Dulith Novaes.

Nouwen, Rick. 2003. *Plural Pronominal Anaphora in Context*. Netherlands Graduate School of Linguistics Dissertations 84. Utrecht: LOT.

———. 2007. "On Dependent Pronouns and Dynamic Semantics." *Journal of Philosophical Logic* 36 (2): 123–54.

Nouwen, Rick, Adrian Brasoveanu, Jan van Eijck, and Albert Visser. 2016. "Dynamic Semantics." *Stanford Encyclopedia of Philosophy*.

van den Berg, M.H. 1996. "Some Aspects of the Internal Structure of Discourse: The Dynamics of Nominal Anaphora." PhD thesis, ILLC, Universiteit van Amsterdam.

van Eijck, Jan. 2001. "Incremental Dynamics." *Journal of Logic Language and Information* 10 (3): 319–51.

van Eijck, Jan, and Rick Nouwen. 2002. "Quantification and Reference in Incremental Processing." *Unpublished Manuscript, CWI, ILLC and UiL-OTS. Presented at the (Preferably) Non-Lexical Semantics Conference*.